Control-Flow Refinement for Complexity Analysis of Probabilistic Programs in KoAT

International Joint Conference on Automated Reasoning 2024

Nils Lommen, Éléanore Meyer, and Jürgen Giesl

Overview

Goal: Infer (upper) time bounds for programs

Overview

Goal: Infer (upper) time bounds for programs

```
while (1 ≤ x ≤ u) do
  if (0 < y) then
    x ← x + 1
  else
    x ← x − 1
  fi
end
```


IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer (upper) time bounds for programs

```
while (1 ≤ x ≤ u) do
  if (0 < y) then
    x ← x + 1
  else
    x ← x − 1
  fi
end
```

- Either always then or else-part executed

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer (upper) time bounds for programs

```
while (1 ≤ x ≤ u) do
  if (0 < y) then
    x ← x + 1
  else
    x ← x − 1
  fi
end
```

- Either always then or else-part executed
  - Control-Flow Refinement (CFR) sorts-out unnecessary paths [Doménech et al.]

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer (upper) time bounds for programs

```
while (1 ≤ x ≤ u) do
  if (0 < y) then
    x ← x + 1
  else
    x ← x − 1
  fi
end
```

- Either always then or else-part executed
  - Control-Flow Refinement (CFR) sorts-out unnecessary paths [Doménech et al.]
  - No linear ranking function as it must be decreasing and increasing in x

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer (upper) time bounds for programs

```
if (0 < y) then
  while (1 ≤ x ≤ u) do
    x ← x + 1
  end
else
  while (1 ≤ x ≤ u) do
    x ← x − 1
  end
fi
```

- Either always then or else-part executed
  - Control-Flow Refinement (CFR) sorts-out unnecessary paths [Doménech et al.]
  - No linear ranking function as it must be decreasing and increasing in x

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer (upper) time bounds for programs

```
if (0 < y) then
  while (1 ≤ x ≤ u) do
    x ← x + 1
  end
else
  while (1 ≤ x ≤ u) do
    x ← x − 1
  end
fi
```

- Either always then or else-part executed
  - Control-Flow Refinement (CFR) sorts-out unnecessary paths [Doménech et al.]
  - No linear ranking function as it must be decreasing and increasing in x
- CFR does not change runtime complexity

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer (upper) time bounds for programs

```
if (0 < y) then
  while (1 ≤ x ≤ u) do
    x ← x + 1
  end
else
  while (1 ≤ x ≤ u) do
    x ← x − 1
  end
fi
```

- Either always then or else-part executed
  - Control-Flow Refinement (CFR) sorts-out unnecessary paths [Doménech et al.]
  - No linear ranking function as it must be decreasing and increasing in x
- CFR does not change runtime complexity
  $\Rightarrow$ Analyze transformed program [SRH'22]

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer (upper) time bounds for programs

```
if (0 < y) then
  while (1 ≤ x ≤ u) do
    x ← x + 1
  end
else
  while (1 ≤ x ≤ u) do
    x ← x − 1
  end
fi
```

- Either always then or else-part executed
  - Control-Flow Refinement (CFR) sorts-out unnecessary paths [Doménech et al.]
  - No linear ranking function as it must be decreasing and increasing in x
- CFR does not change runtime complexity
  $\Rightarrow$ Analyze transformed program [SRH'22]
- Time bound: $\max(u, u) = u$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer (upper) expected time bounds for probabilistic programs

```
while (1 ≤ x ≤ u) do
  if (0 < y) then
    x ← x + 1
  else
    x ← x − 1
  fi
end
```


IJCAR '24

Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer (upper) expected time bounds for probabilistic programs

```
while (1 ≤ x ≤ u) do
  if (0 < y) then
    x ← x + 1 ⊕_{1/2} x
  else
    x ← x − 1
  fi
end
```


IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer (upper) expected time bounds for probabilistic programs

```
while (1 ≤ x ≤ u) do
  if (0 < y) then
    x ← x + 1 ⊕_{1/2} x
  else
    x ← x − 1
  fi
end
```

- Either always then or else-part executed
  - Control-Flow Refinement (CFR) sorts-out unnecessary paths
  - No linear probabilistic ranking function as it must be decreasing and increasing in x

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer (upper) expected time bounds for probabilistic programs

```
if (0 < y) then
  while (1 ≤ x ≤ u) do
    x ← x + 1 ⊕_{1/2} x
  end
else
  while (1 ≤ x ≤ u) do
    x ← x − 1
  end
fi
```

- Either always then or else-part executed
  - Control-Flow Refinement (CFR) sorts-out unnecessary paths
  - No linear probabilistic ranking function as it must be decreasing and increasing in x

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer (upper) expected time bounds for probabilistic programs

```
if (0 < y) then
  while (1 ≤ x ≤ u) do
    x ← x + 1 ⊕_{1/2} x
  end
else
  while (1 ≤ x ≤ u) do
    x ← x − 1
  end
fi
```

- Either always then or else-part executed
  - Control-Flow Refinement (CFR) sorts-out unnecessary paths
  - No linear probabilistic ranking function as it must be decreasing and increasing in x
- CFR does not change expected runtime complexity
  $\Rightarrow$ Analyze transformed program

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer (upper) expected time bounds for probabilistic programs

```
if (0 < y) then
  while (1 ≤ x ≤ u) do
    x ← x + 1 ⊕_{1/2} x
  end
else
  while (1 ≤ x ≤ u) do
    x ← x − 1
  end
fi
```

- Either always then or else-part executed
  - Control-Flow Refinement (CFR) sorts-out unnecessary paths
  - No linear probabilistic ranking function as it must be decreasing and increasing in $\mathtt{x}$
- CFR does not change expected runtime complexity
  $\Rightarrow$ Analyze transformed program
- Expected time bound: $\max(2 \cdot \mathtt{u}, \mathtt{u}) = 2 \cdot \mathtt{u}$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer (upper) expected time bounds for probabilistic programs

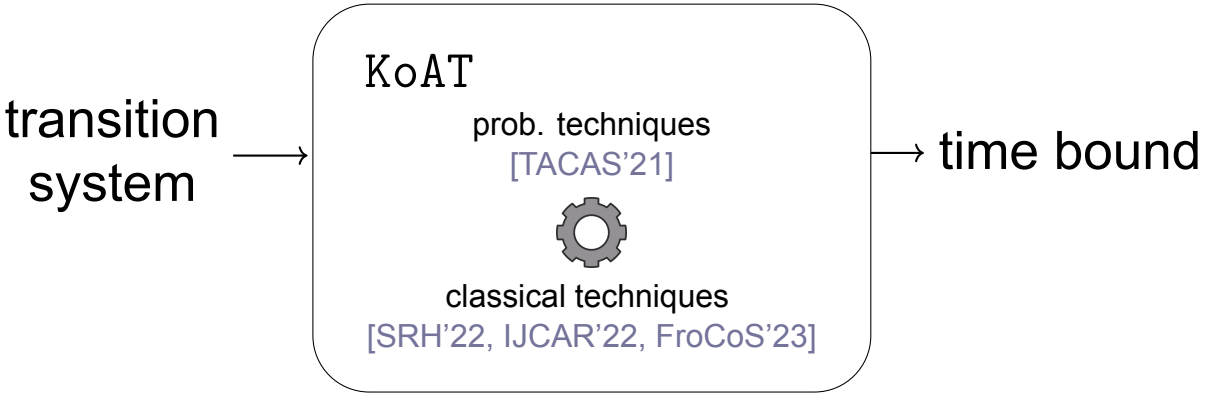
```
if (0 < y) then
  while (1 ≤ x ≤ u) do
    x ← x + 1 ⊕_{1/2} x
  end
else
  while (1 ≤ x ≤ u) do
    x ← x − 1
  end
fi
```

- Either always then or else-part executed
  - Control-Flow Refinement (CFR) sorts-out unnecessary paths
  - No linear probabilistic ranking function as it must be decreasing and increasing in $\mathtt{x}$
- CFR does not change expected runtime complexity
  $\Rightarrow$ Analyze transformed program
- Expected time bound: $\max(2 \cdot \mathtt{u}, \mathtt{u}) = 2 \cdot \mathtt{u}$
- Independent subsystem in modular complexity analysis tool KoAT

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

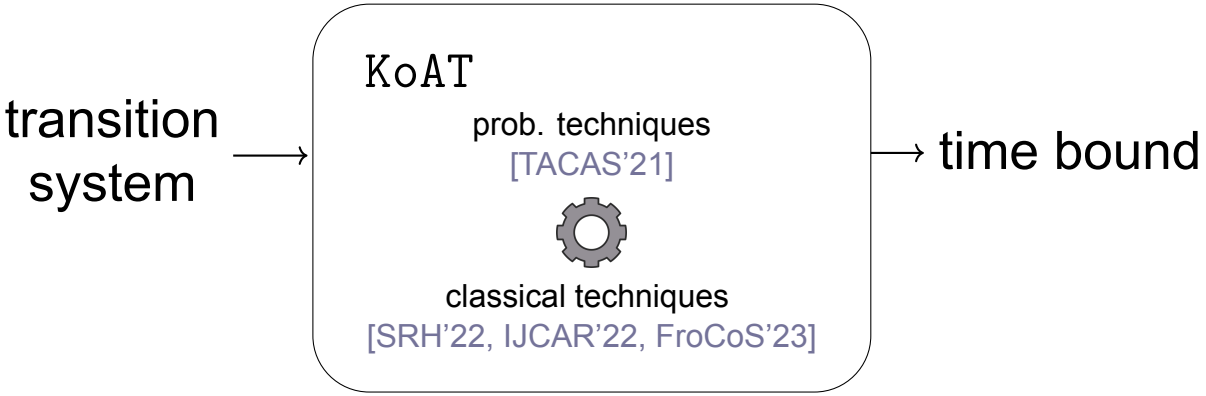
Overview

Goal: Infer upper runtime bounds for (probabilistic) programs



Overview

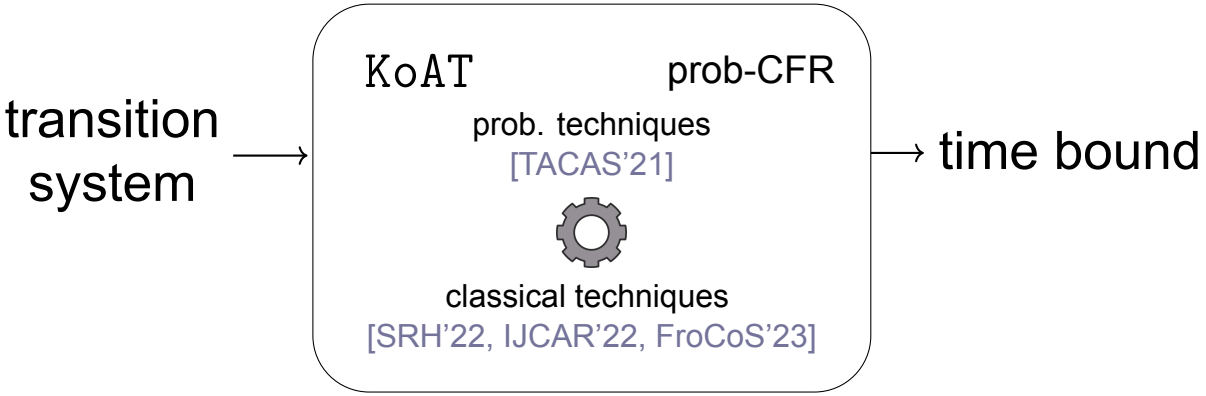
Goal: Infer upper runtime bounds for (probabilistic) programs



Contributions:

Overview

Goal: Infer upper runtime bounds for (probabilistic) programs



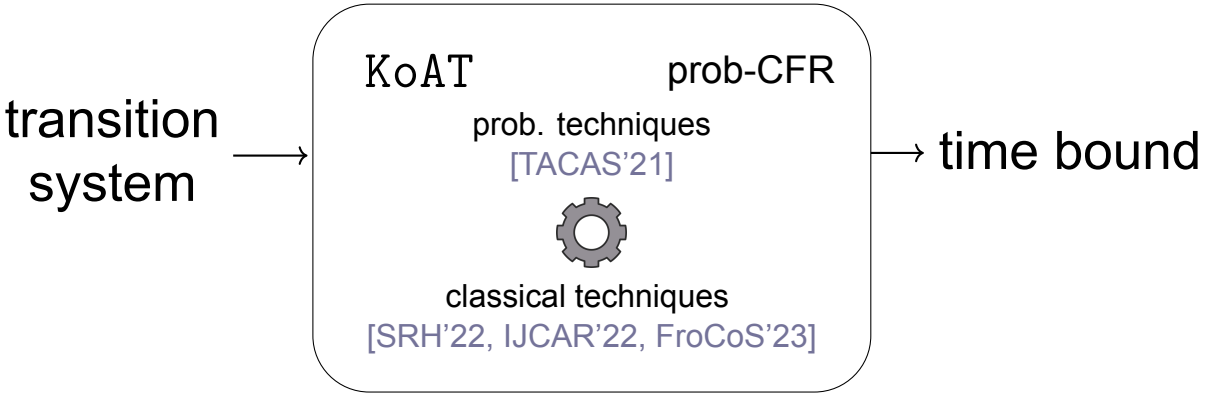
Contributions:

- Integrate *probabilistic* (modular) control-flow refinement

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer upper runtime bounds for (probabilistic) programs



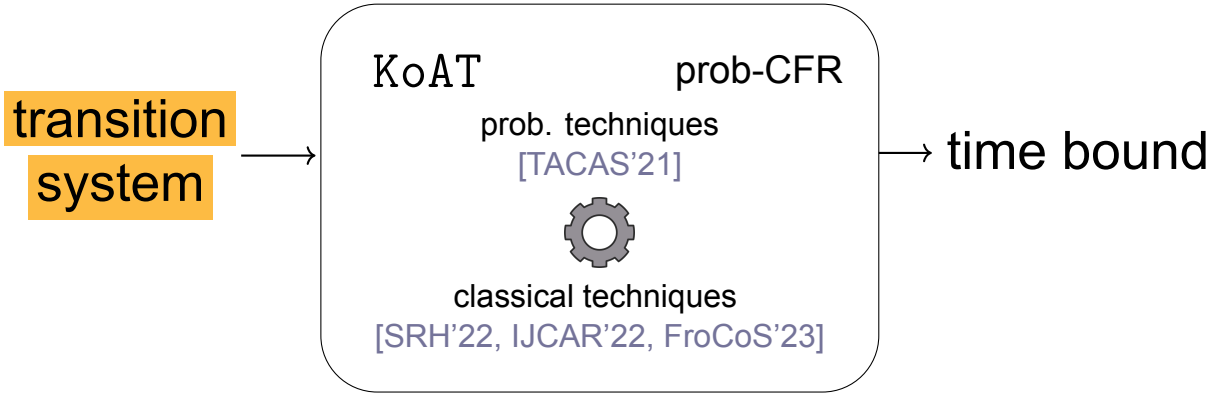
Contributions:

- Integrate *probabilistic* (modular) control-flow refinement
- Provide implementation in complexity analysis tool KoAT

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer upper runtime bounds for (probabilistic) programs



Contributions:

- Integrate *probabilistic* (modular) control-flow refinement
- Provide implementation in complexity analysis tool KoAT

Modelling Programs as (Integer) Transition Systems

CFR-Algorithm works on ITS

```
while (1 ≤ x ≤ u) do
  if (0 < y) then
    x ← x + 1 ⊕_{1/2} x
  else
    x ← x − 1
  fi
end
```


IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Modelling Programs as (Integer) Transition Systems

CFR-Algorithm works on ITS

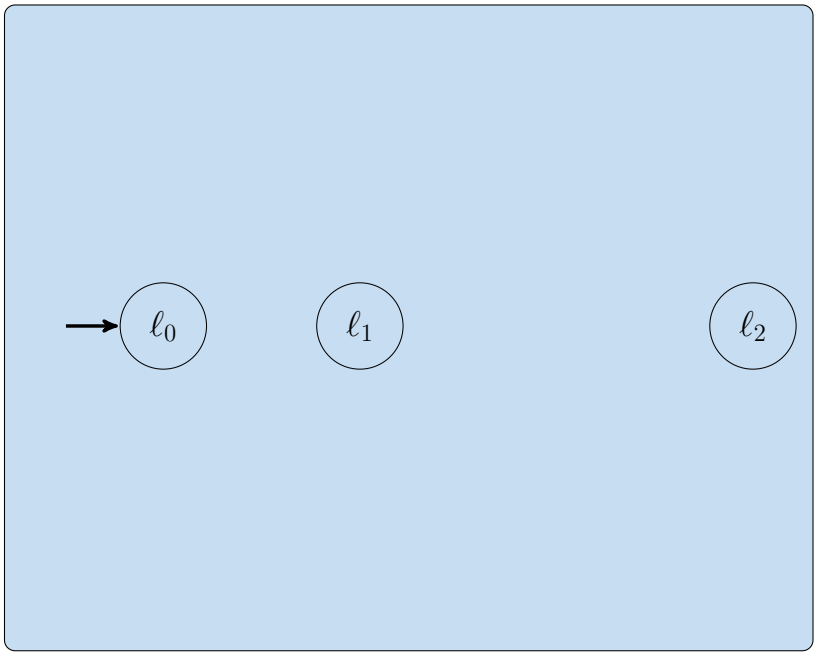
```
while (1 ≤ x ≤ u) do
  if (0 < y) then
    x ← x + 1 ⊕_{1/2} x
  else
    x ← x − 1
  fi
end
```

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Modelling Programs as (Integer) Transition Systems

CFR-Algorithm works on ITS

```
while (1 ≤ x ≤ u) do
  if (0 < y) then
    x ← x + 1 ⊕_{1/2} x
  else
    x ← x − 1
  fi
end
```

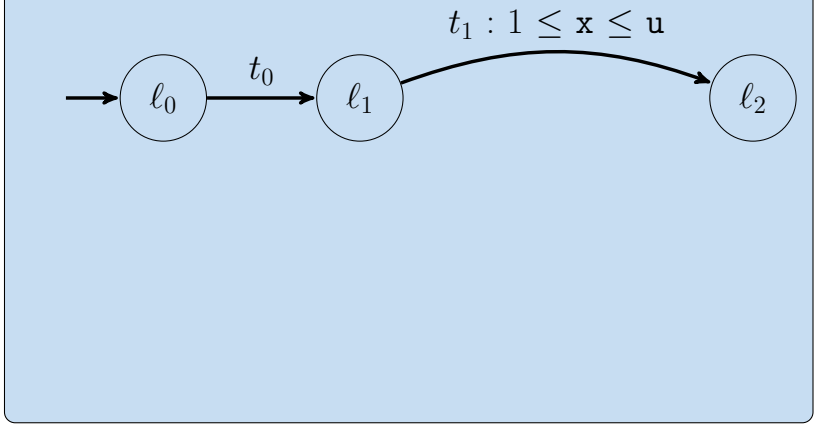


IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Modelling Programs as (Integer) Transition Systems

CFR-Algorithm works on ITS

```
while (1 ≤ x ≤ u) do
  if (0 < y) then
    x ← x + 1 ⊕_{1/2} x
  else
    x ← x − 1
  fi
end
```



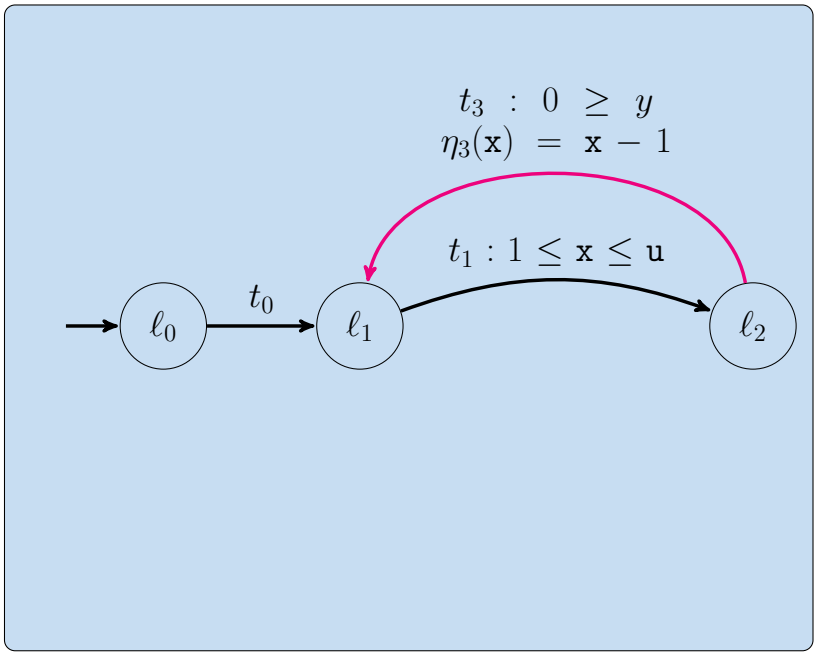
- Guards `true` and updates $\eta(x) = x$ were omitted

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Modelling Programs as (Integer) Transition Systems

CFR-Algorithm works on ITS

```
while (1 ≤ x ≤ u) do
  if (0 < y) then
    x ← x + 1 ⊕_{1/2} x
  else
    x ← x − 1
  fi
end
```



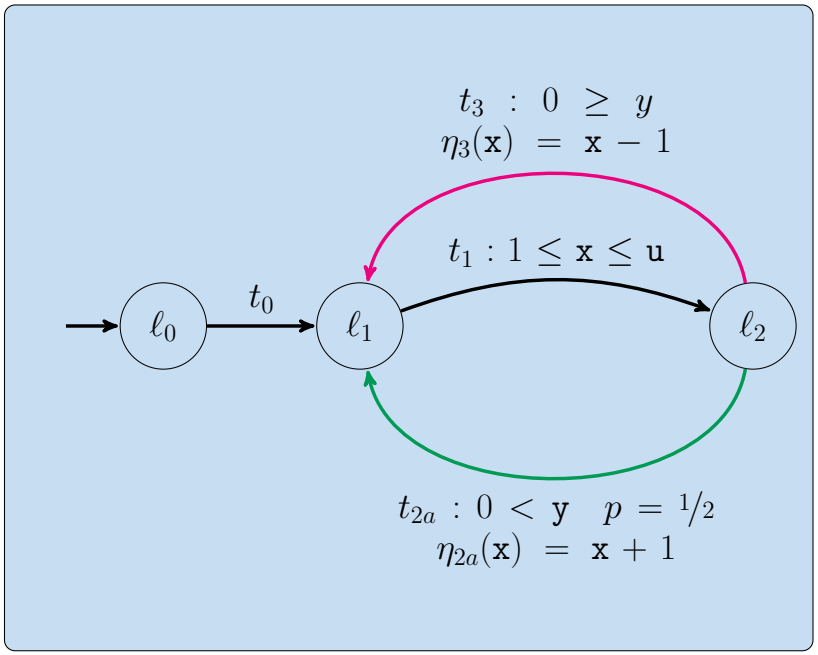
- Guards `true` and updates $\eta(x) = x$ were omitted

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Modelling Programs as (Integer) Transition Systems

CFR-Algorithm works on ITS

```
while (1 ≤ x ≤ u) do
  if (0 < y) then
    x ← x + 1 ⊕_{1/2} x
  else
    x ← x − 1
  fi
end
```



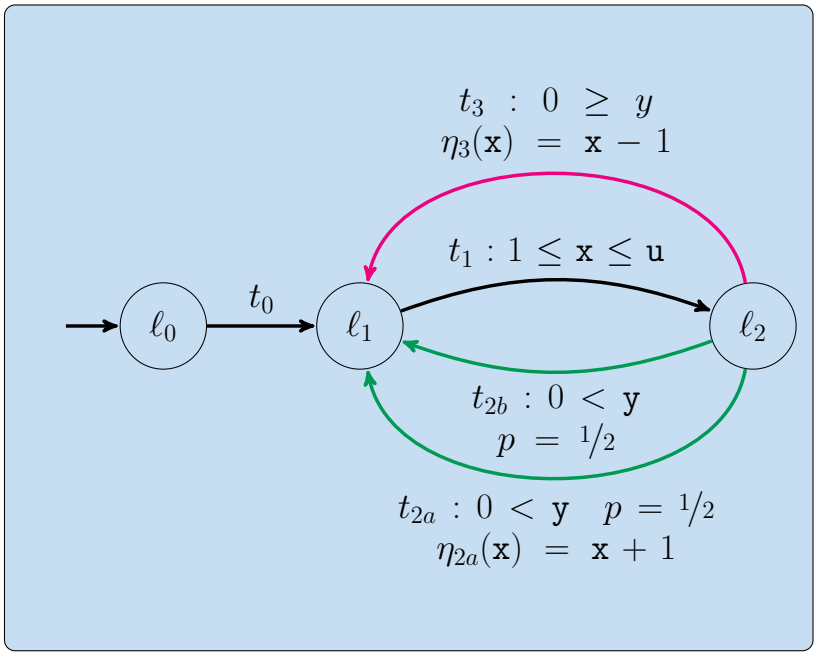
- Guards `true` and updates $\eta(x) = x$ were omitted

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Modelling Programs as (Integer) Transition Systems

CFR-Algorithm works on ITS

```
while (1 ≤ x ≤ u) do
  if (0 < y) then
    x ← x + 1 ⊕_{1/2} x
  else
    x ← x − 1
  fi
end
```



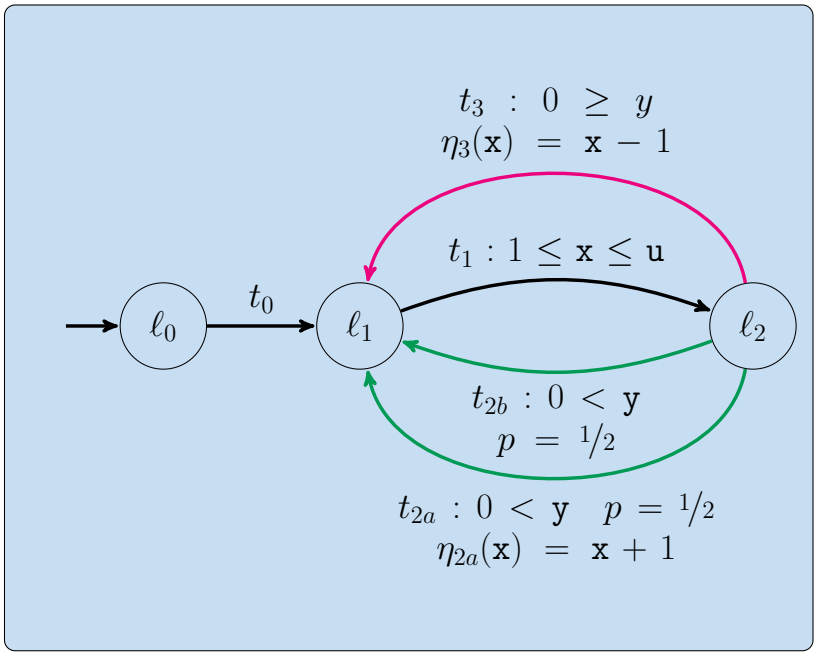
- Guards `true` and updates $\eta(x) = x$ were omitted

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Modelling Programs as (Integer) Transition Systems

CFR-Algorithm works on ITS

```
while (1 ≤ x ≤ u) do
  if (0 < y) then
    x ← x + 1 ⊕_{1/2} x
  else
    x ← x − 1
  fi
end
```

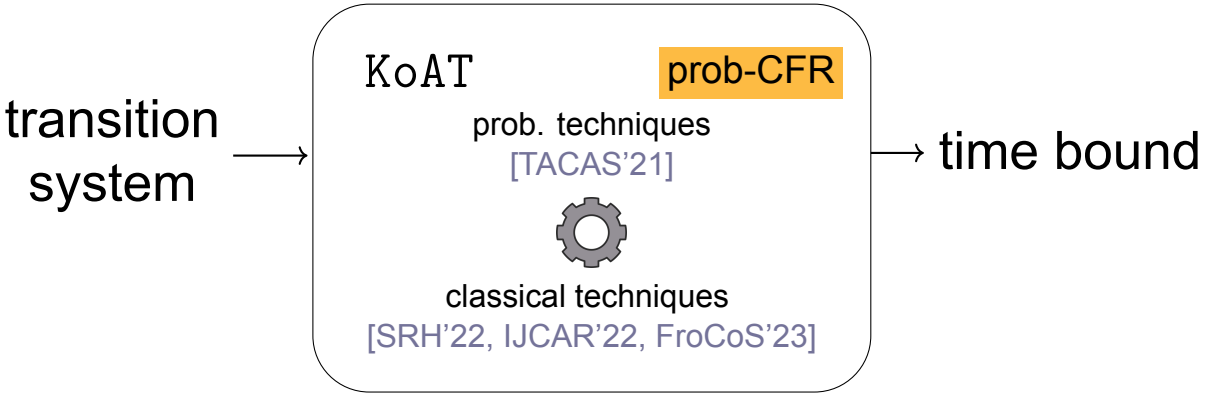


- Guards `true` and updates $\eta(x) = x$ were omitted
- Goal of CFR: Separate strongly connected components

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

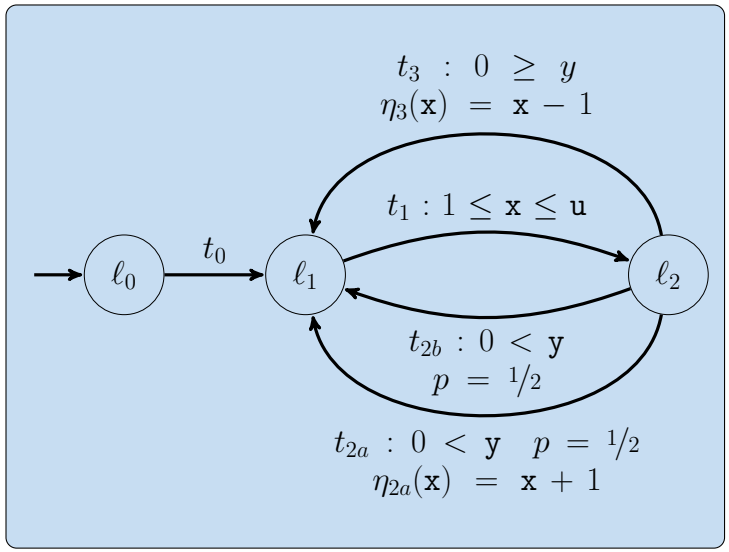
Overview

Goal: Infer upper runtime bounds for (probabilistic) programs



CFR via Partial Evaluation

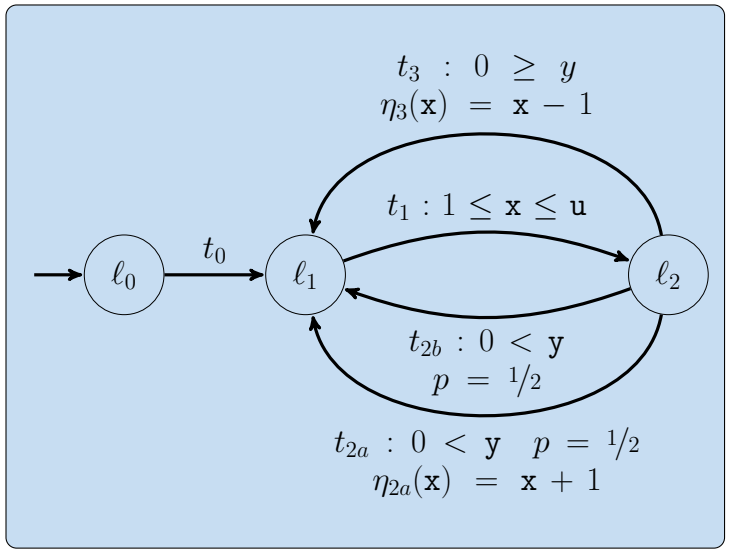
Idea: Use guards and updates to generate refined locations

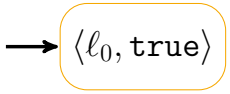


IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations

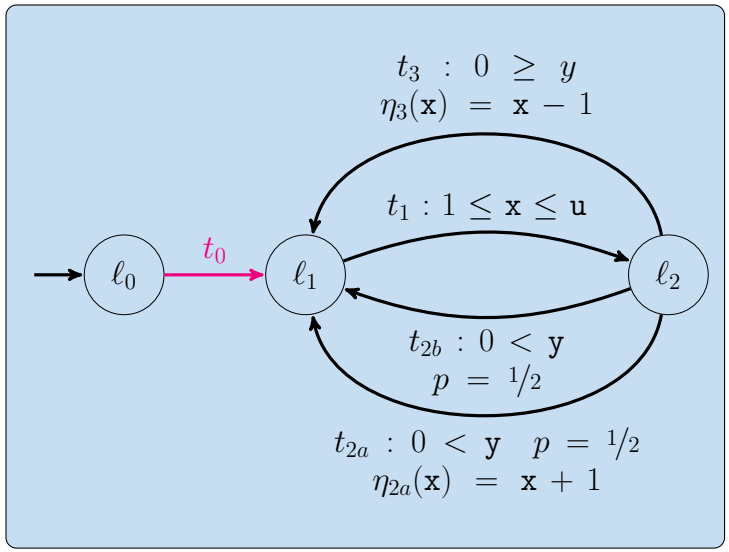


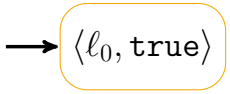


IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations



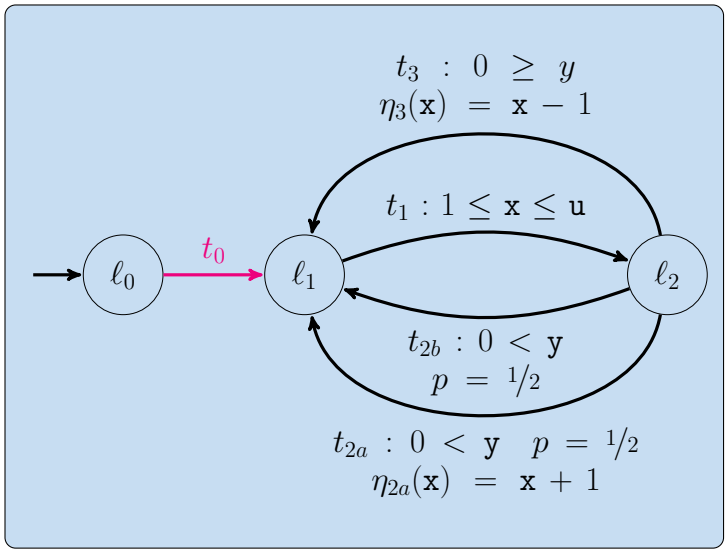


Apply $t_0 : \mathtt{true}$:

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations



$\langle \ell_0, \mathtt{true} \rangle$

Apply $t_0 : \mathtt{true}$:

guard pre. label

$\mathtt{true}, \ \mathtt{true}$

update

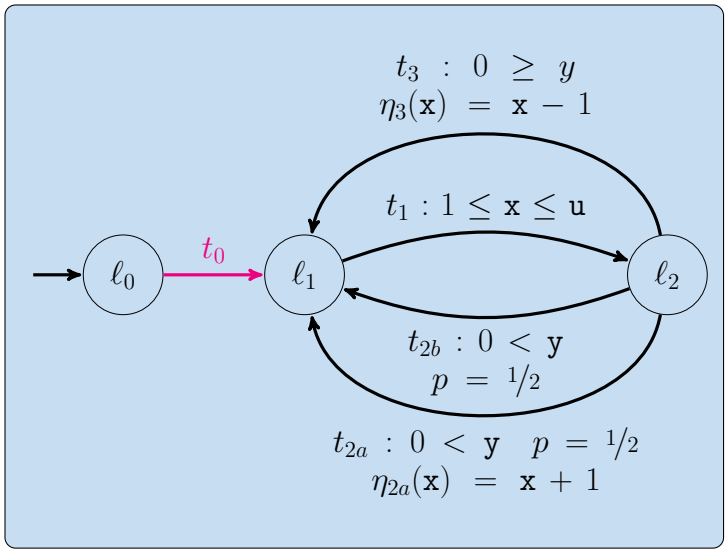
$\longrightarrow$

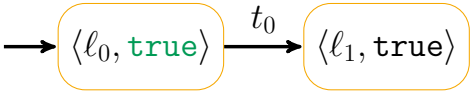
$\mathtt{true}$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations





Apply $t_0 : \mathtt{true}$:

guard pre. label

$\mathtt{true}, \ \mathtt{true}$

update

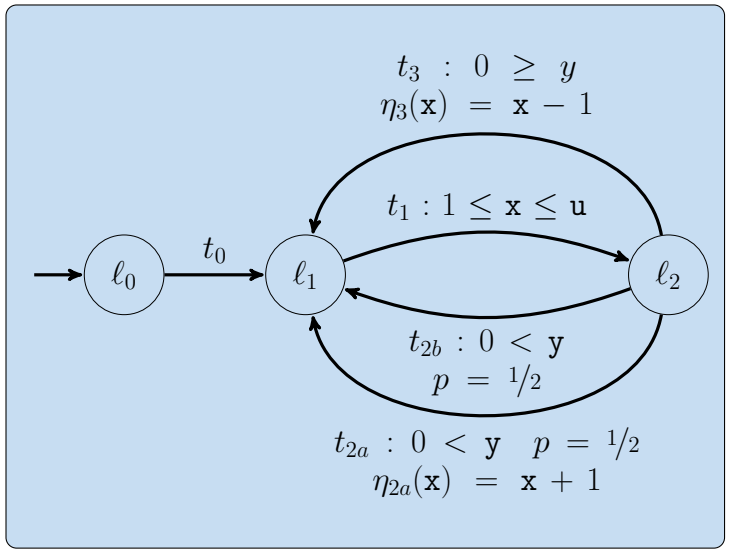
$\longrightarrow$

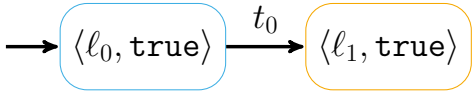
$\mathtt{true}$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations

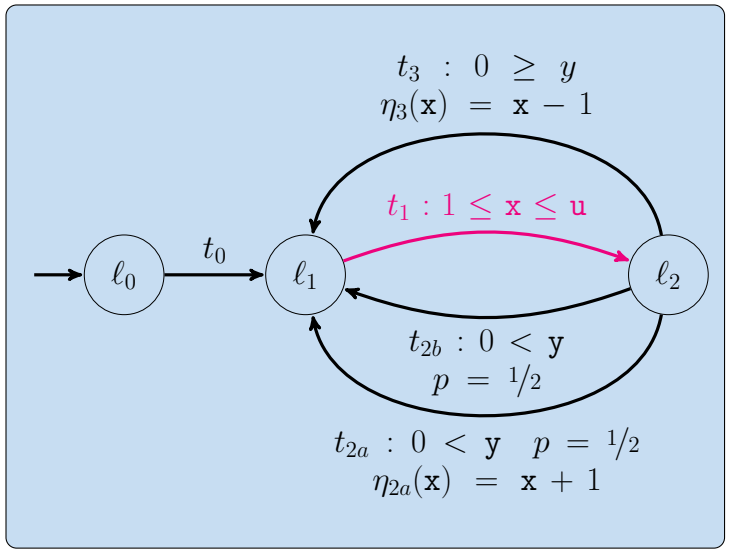


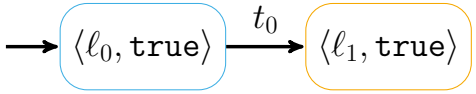


IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations



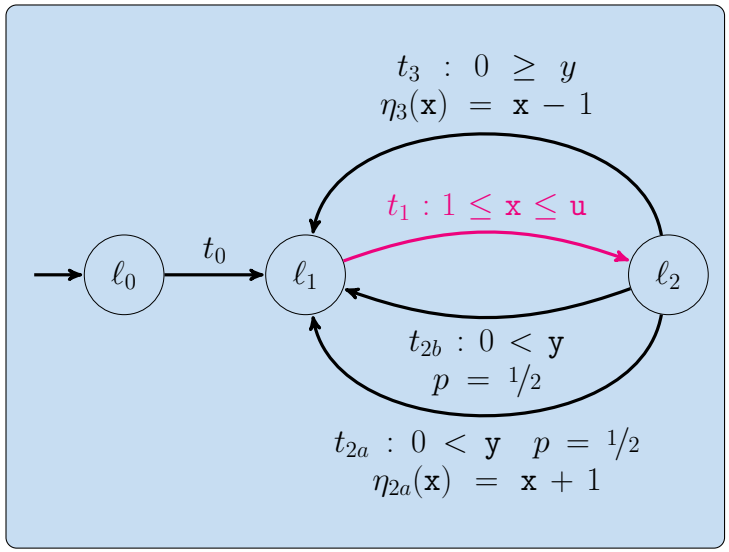


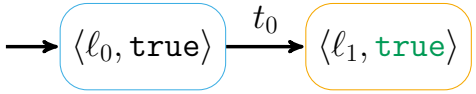
Apply $t_1 : 1 \leq \mathtt{x} \leq \mathtt{u}$:

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations





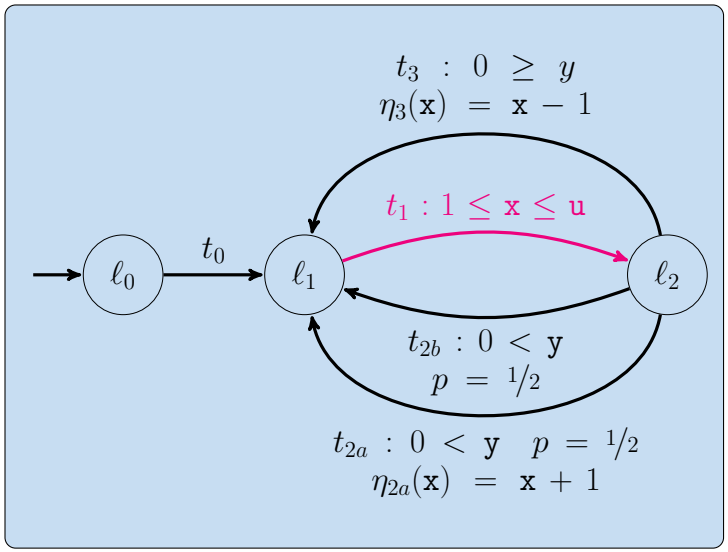
Apply $t_1 : 1 \leq \mathtt{x} \leq \mathtt{u}$:

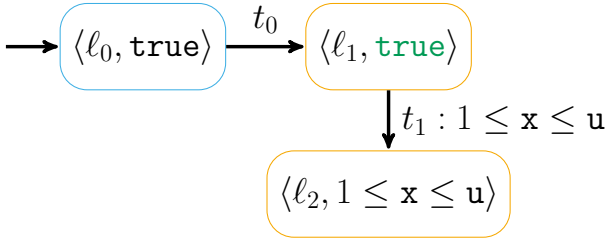
$$\overbrace{1 \leq \mathtt{x} \leq \mathtt{u}}^{\text{guard}}, \overbrace{\mathtt{true}}^{\text{pre. label}} \xrightarrow{\text{update}} 1 \leq \mathtt{x} \leq \mathtt{u}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations





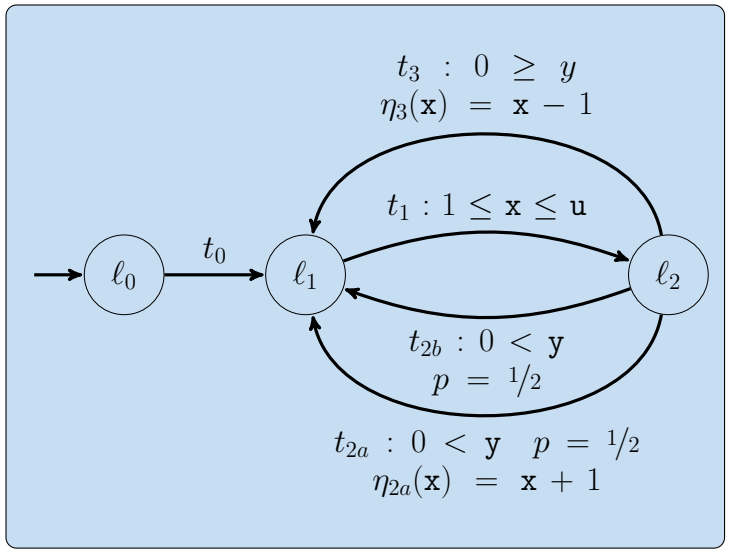
Apply $t_1 : 1 \leq \mathtt{x} \leq \mathtt{u}$:

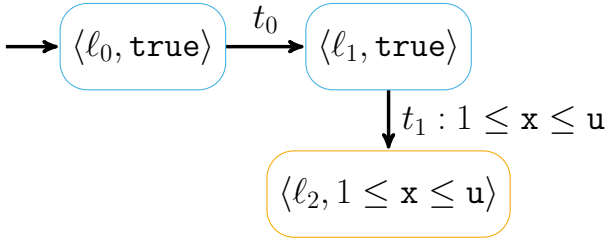
$$\overbrace{1 \leq \mathtt{x} \leq \mathtt{u}}^{\text{guard}}, \overbrace{\mathtt{true}}^{\text{pre. label}} \xrightarrow{\text{update}} 1 \leq \mathtt{x} \leq \mathtt{u}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations

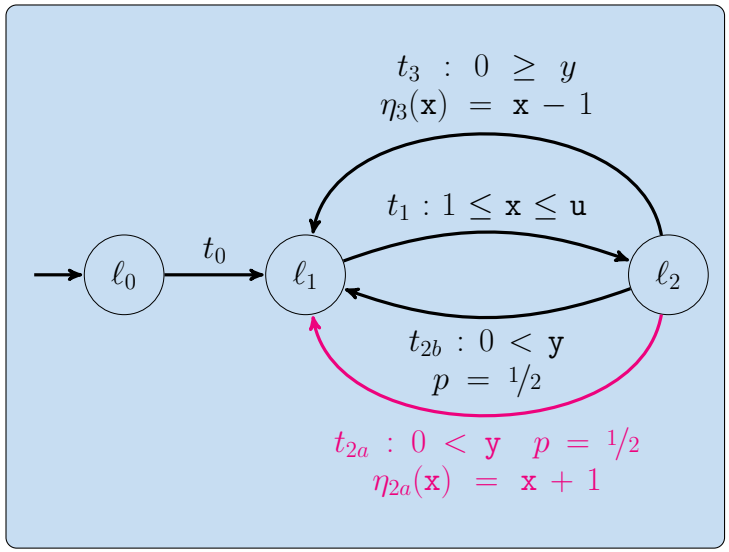


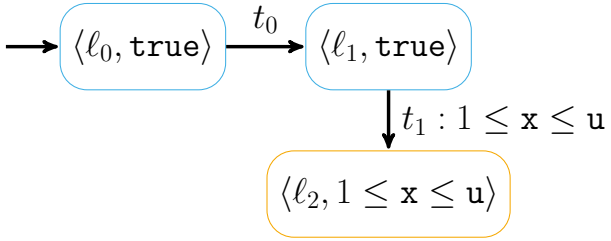


IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations



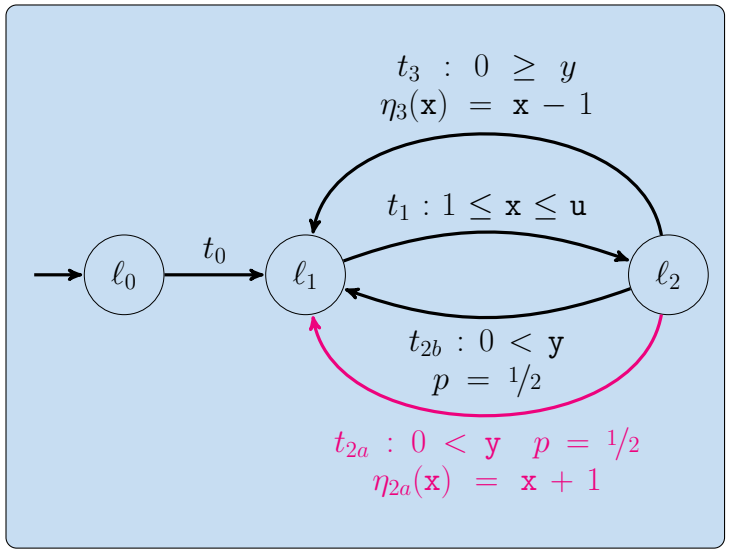


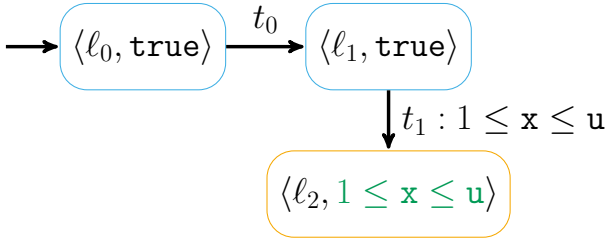
Apply $t_{2a} : 0 < \mathtt{y}$:

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations





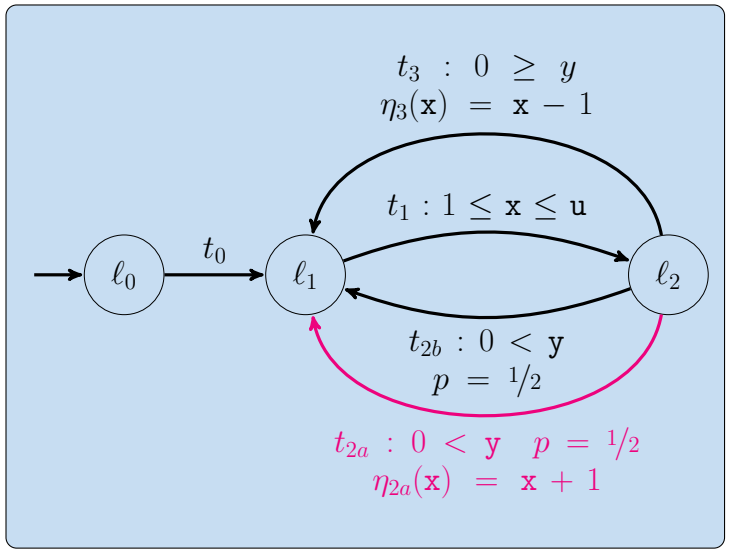
Apply $t_{2a} : 0 < \mathtt{y}$:

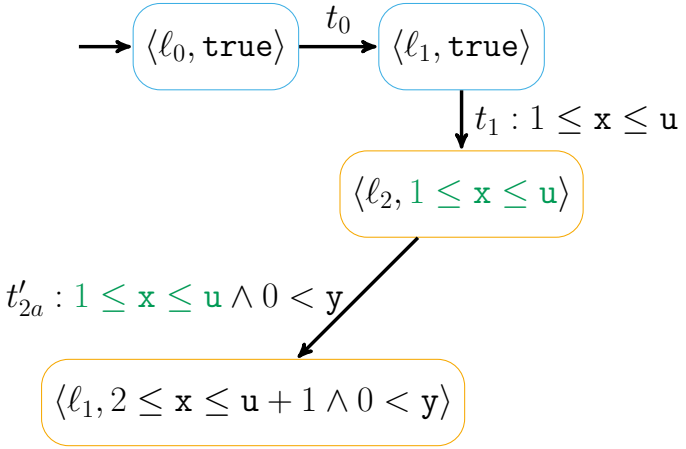
$$\overbrace{0 < \mathtt{y}}^{\text{guard}}, \overbrace{1 \leq \mathtt{x} \leq \mathtt{u}}^{\text{pre. label}} \xrightarrow{\text{update}} 2 \leq \mathtt{x} \leq \mathtt{u} + 1 \wedge 0 < \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations





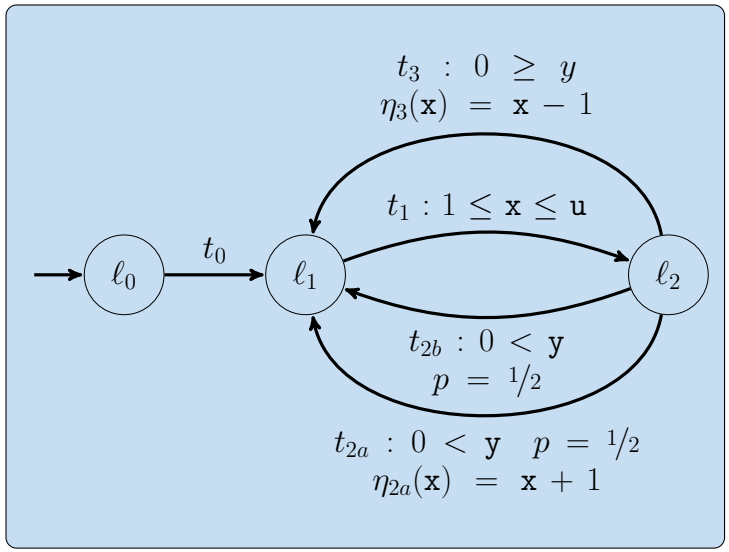
Apply $t_{2a} : 0 < \mathtt{y}$:

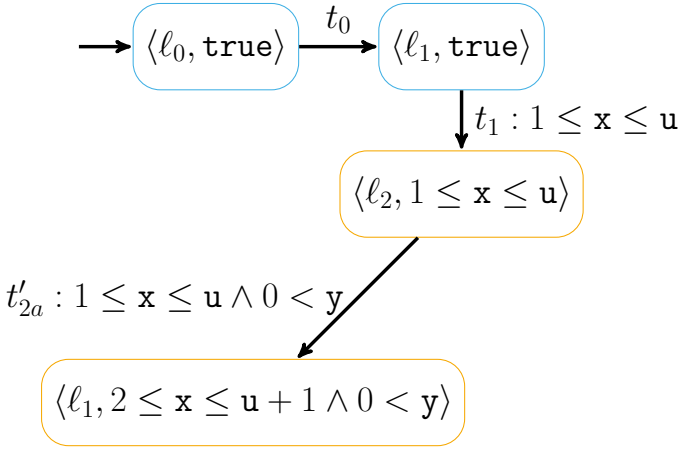
$$\underbrace{0 < \mathtt{y}}_{\text{guard}}, \underbrace{1 \leq \mathtt{x} \leq \mathtt{u}}_{\text{pre. label}} \xrightarrow{\text{update}} 2 \leq \mathtt{x} \leq \mathtt{u} + 1 \land 0 < \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations

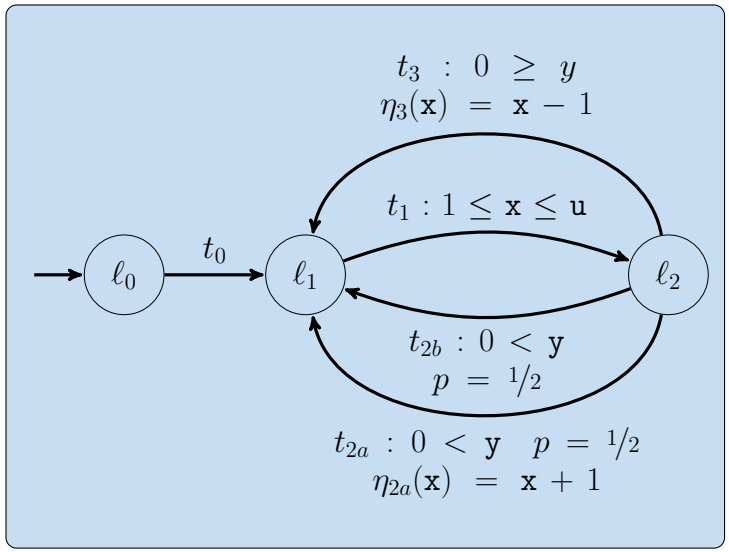


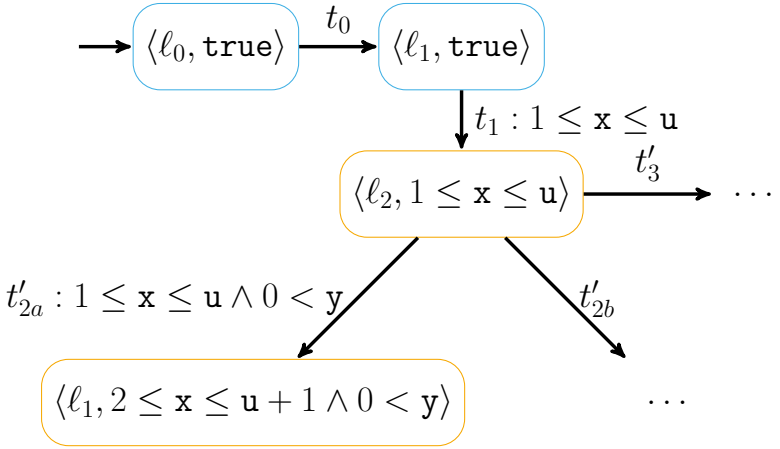


IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations

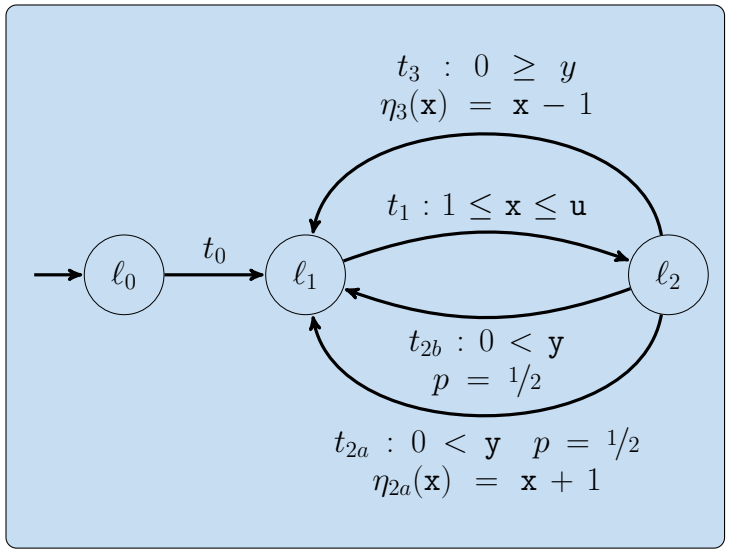


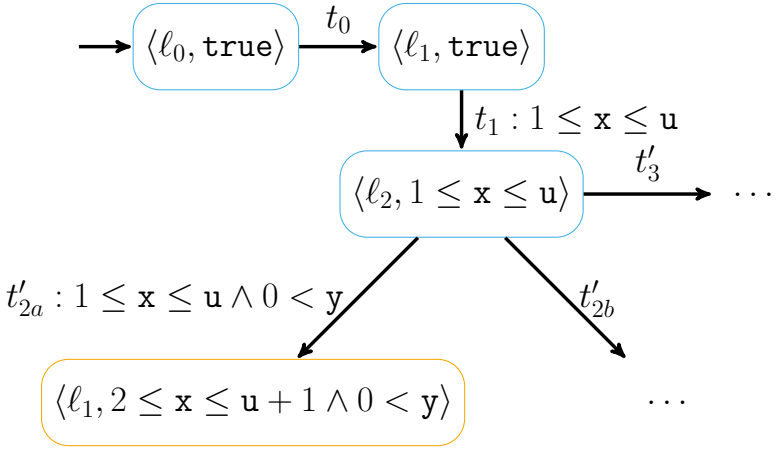


IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations

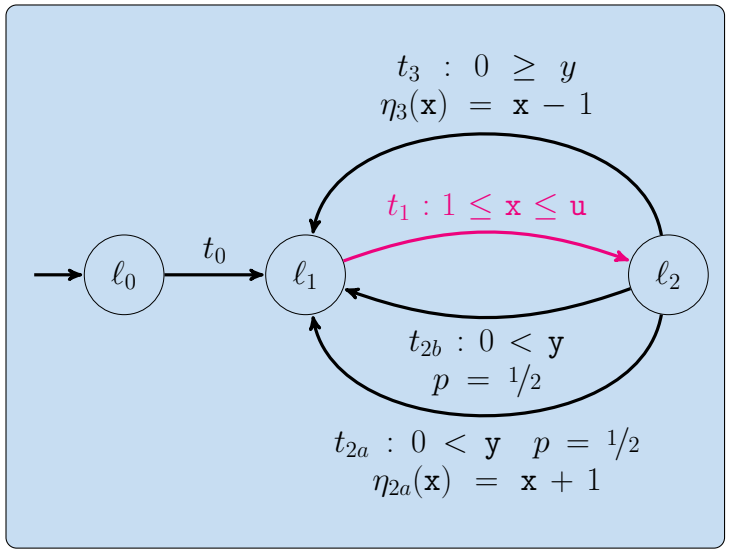


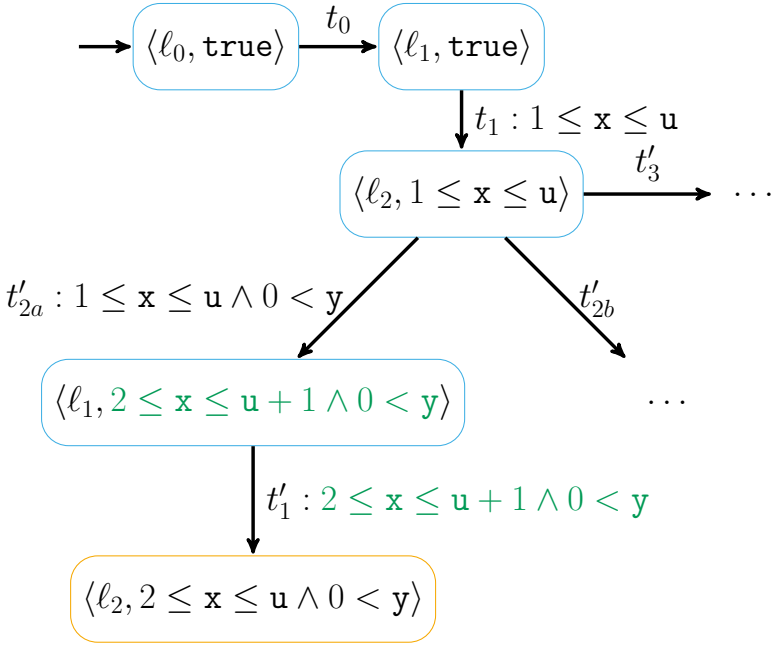


IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations





Apply $t_1 : 1 \leq \mathtt{x} \leq \mathtt{u}$:

$$\overbrace{1 \leq \mathtt{x} \leq \mathtt{u}}^{\text{guard}}, \overbrace{2 \leq \mathtt{x} \leq \mathtt{u} + 1 \land 0 < \mathtt{y}}^{\text{pre. label}}$$

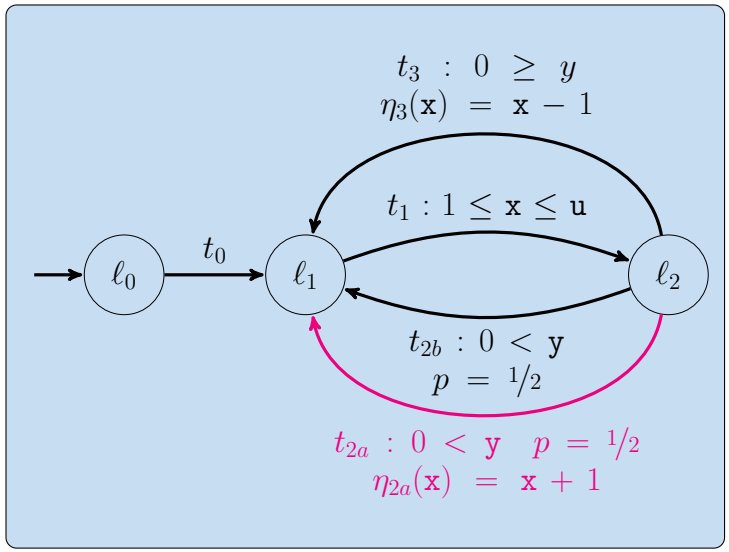
$$\xrightarrow{\text{update}}$$

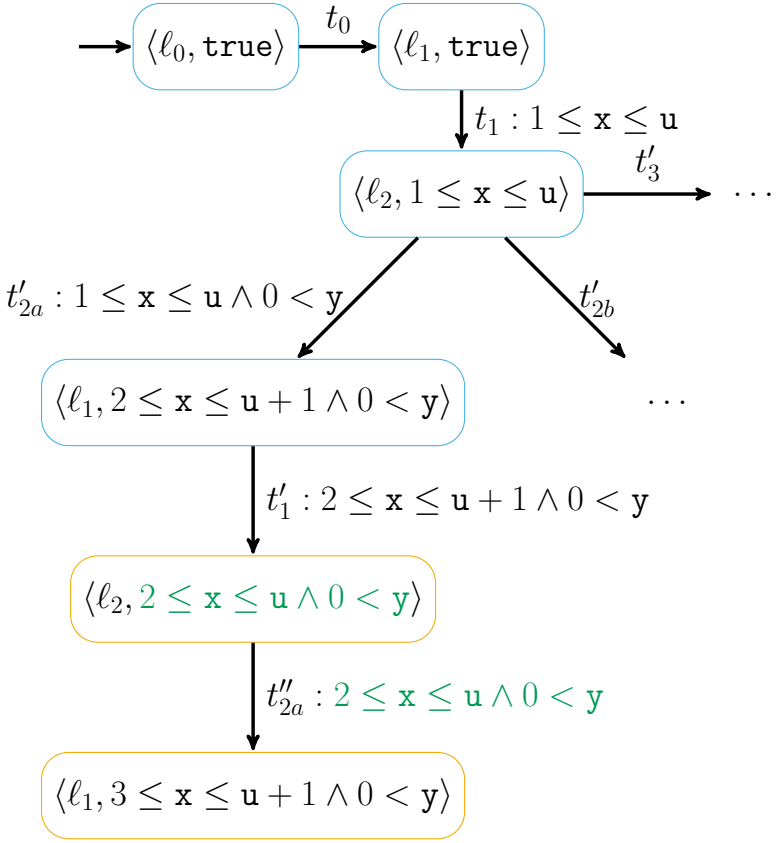
$$2 \leq \mathtt{x} \leq \mathtt{u} \land 0 < \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations





Apply $t_{2a} : 0 < \mathtt{y}$:

guard: $0 < \mathtt{y}$, pre. label: $2 \leq \mathtt{x} \leq \mathtt{u} \land 0 < \mathtt{y}$

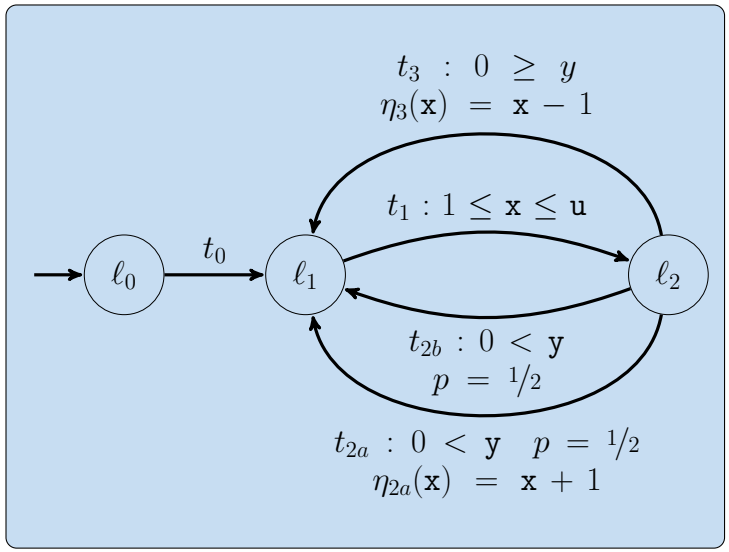
update $\longrightarrow$

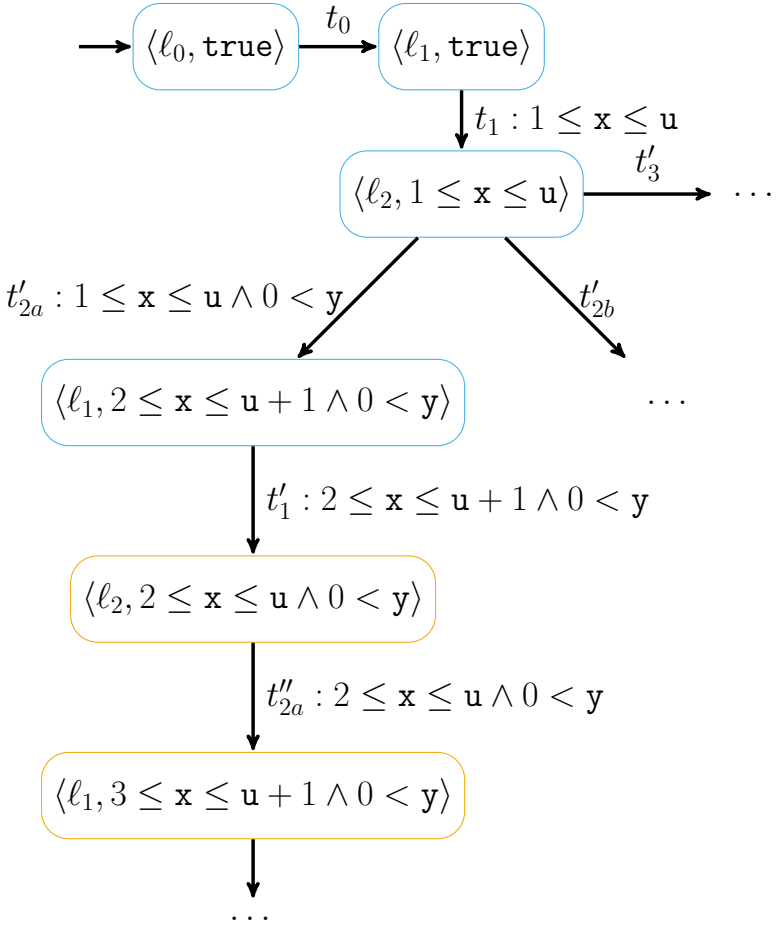
$$3 \leq \mathtt{x} \leq \mathtt{u} + 1 \land 0 < \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation

Idea: Use guards and updates to generate refined locations

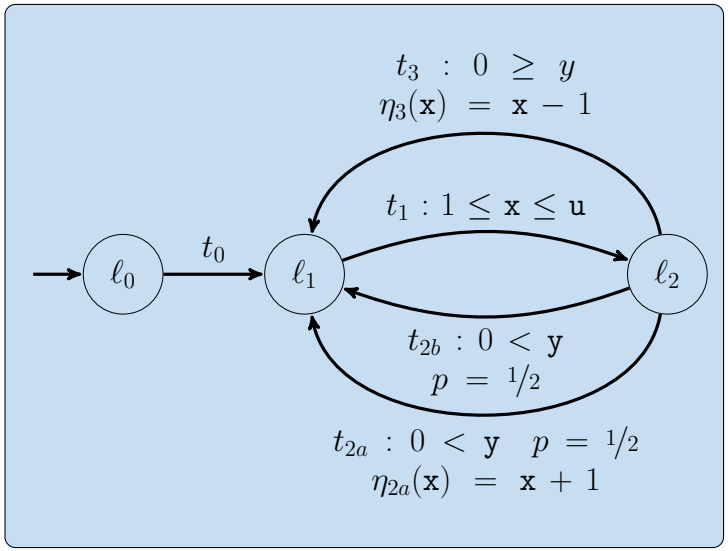




IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

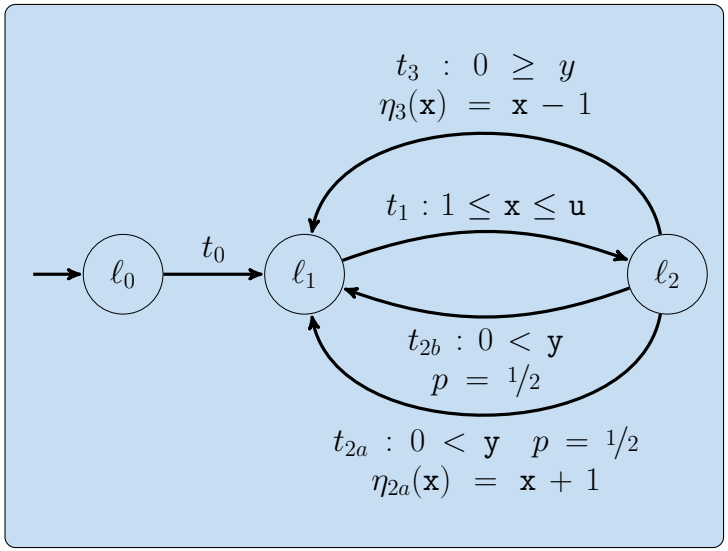
Idea: Use guards and updates to generate refined locations



IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations

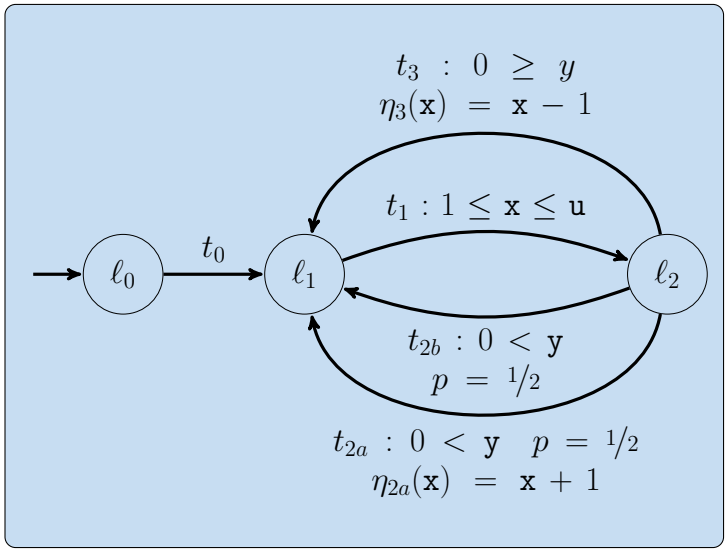


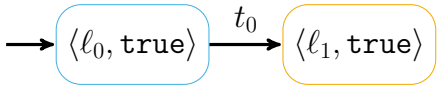
- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations



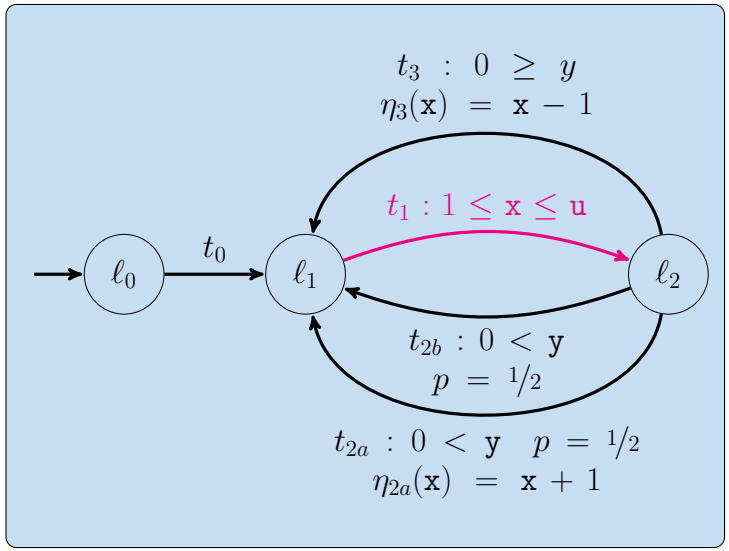


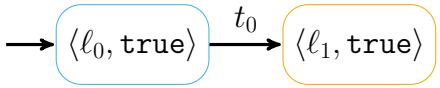
- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





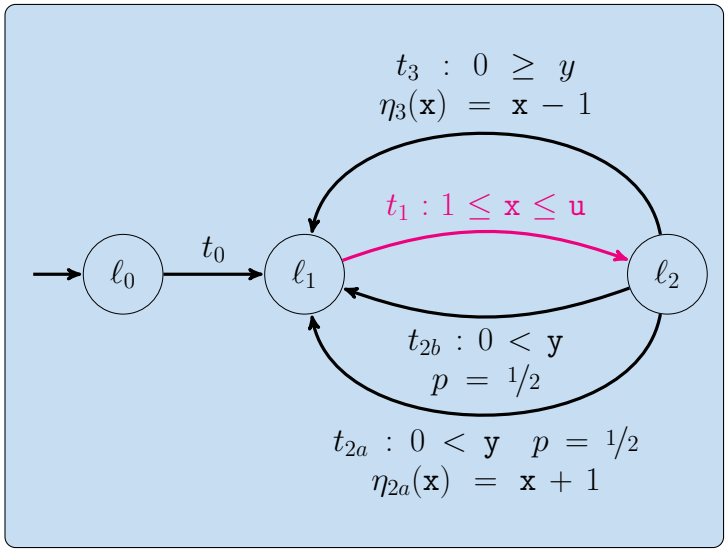
- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

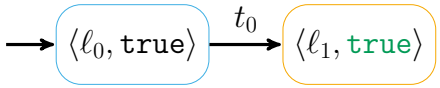
Apply t_1:

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

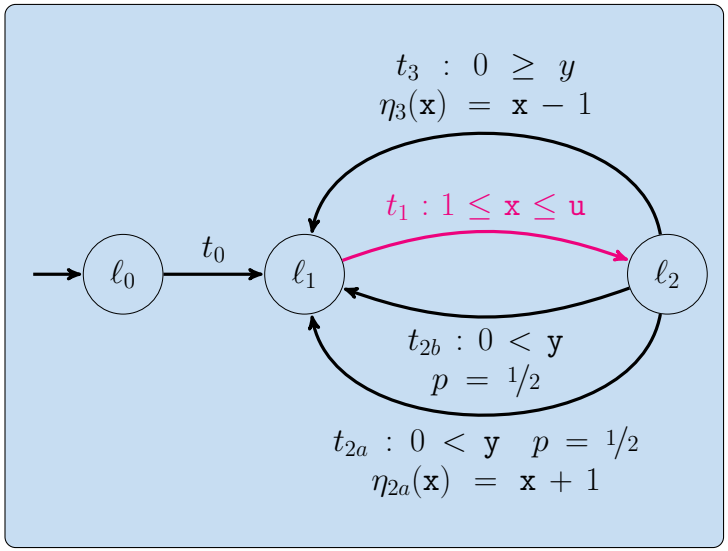
Apply t_1:

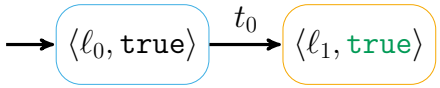
$$\overbrace{1 \leq \mathtt{x} \leq \mathtt{u}}^{\text{guard}}, \overbrace{\mathtt{true}}^{\text{pre. label}} \xrightarrow{\text{update}} 1 \leq \mathtt{x} \leq \mathtt{u}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

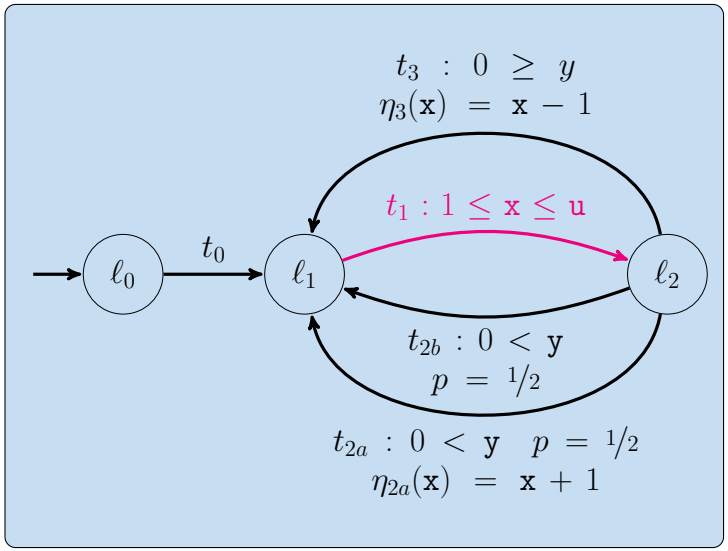
Apply t_1:

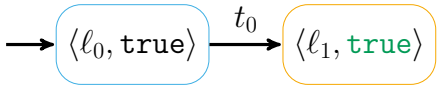
$$\overbrace{1 \leq \mathtt{x} \leq \mathtt{u}}^{\text{guard}}, \overbrace{\mathtt{true}}^{\text{pre. label}} \xrightarrow{\text{update}} 1 \leq \mathtt{x} \leq \mathtt{u} \qquad 1 \leq \mathtt{x} \leq \mathtt{u} \not\models 0 \geq \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

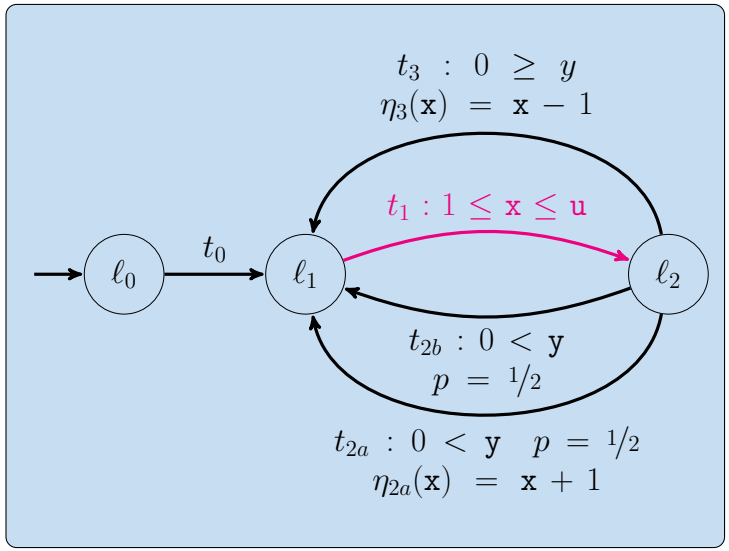
Apply t_1:

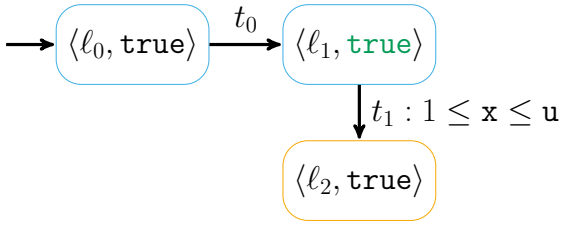
$$\overbrace{1 \leq \mathtt{x} \leq \mathtt{u}}^{\text{guard}}, \overbrace{\mathtt{true}}^{\text{pre. label}} \xrightarrow{\text{update}} 1 \leq \mathtt{x} \leq \mathtt{u} \qquad 1 \leq \mathtt{x} \leq \mathtt{u} \not\models 0 < \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

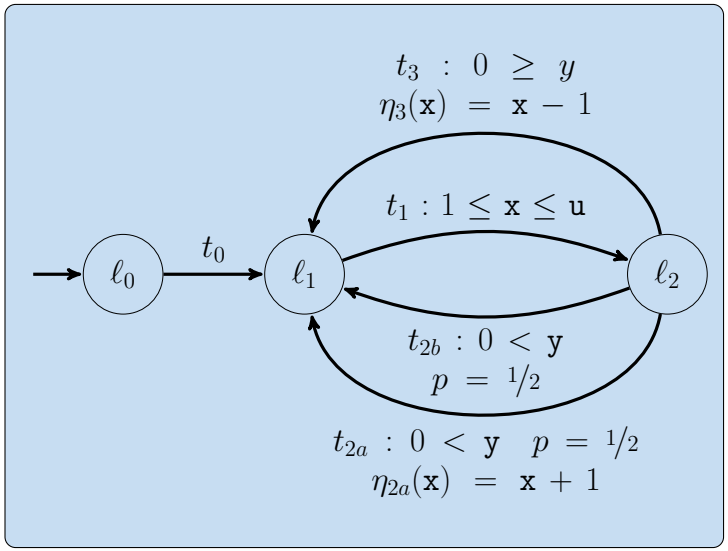
Apply t_1:

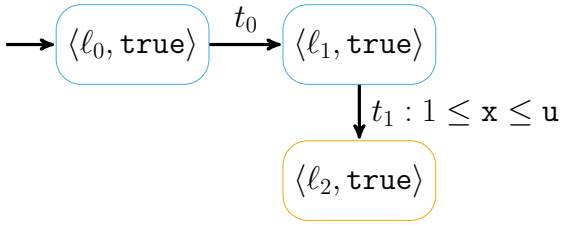
$$\overbrace{1 \leq \mathtt{x} \leq \mathtt{u}}^{\text{guard}}, \overbrace{\mathtt{true}}^{\text{pre. label}} \xrightarrow{\text{update}} 1 \leq \mathtt{x} \leq \mathtt{u} \qquad 1 \leq \mathtt{x} \leq \mathtt{u} \not\models 0 < \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





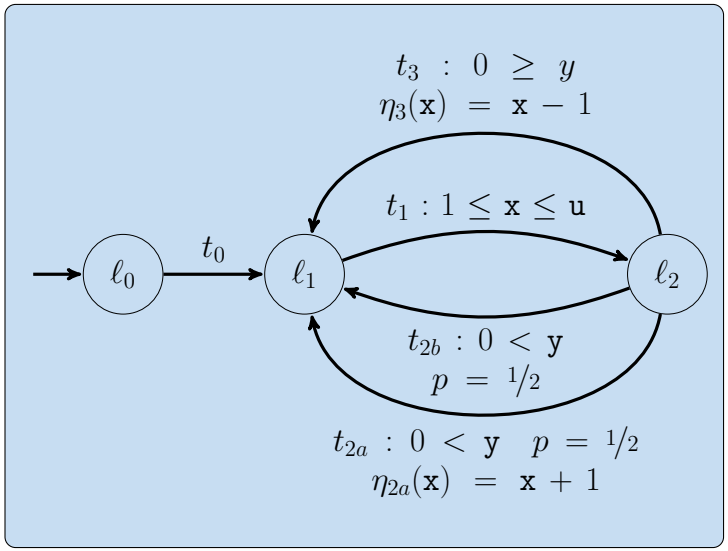
- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

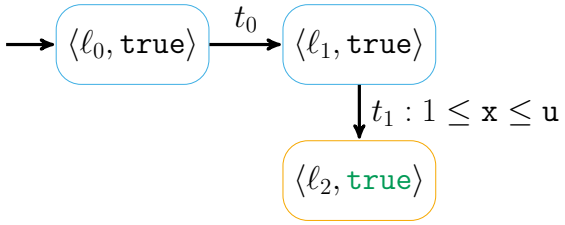
Apply t_{2a}:

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

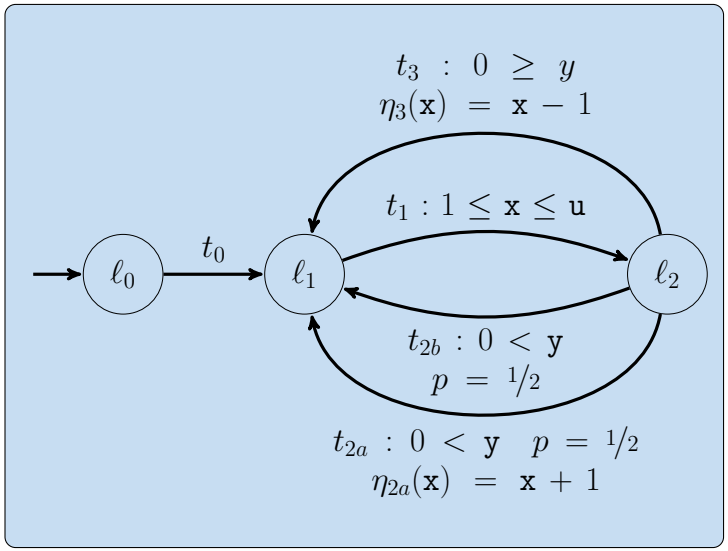
Apply t_{2a}:

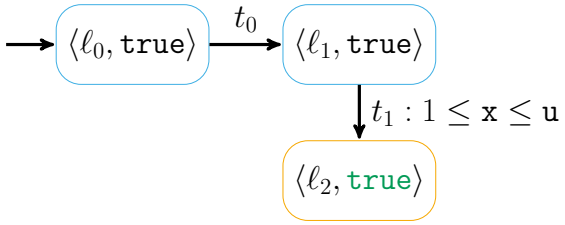
$$\overbrace{0 < \mathtt{y}}^{\text{guard}}, \overbrace{\mathtt{true}}^{\text{pre. label}} \xrightarrow{\text{update}} 0 < \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

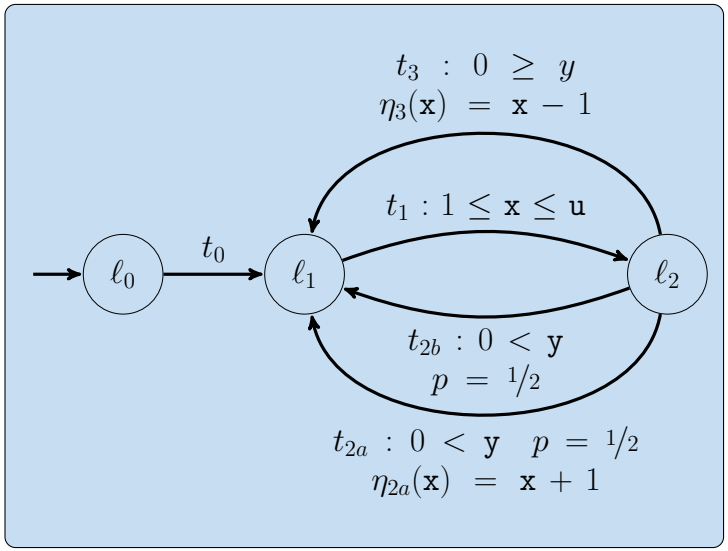
Apply t_{2a}:

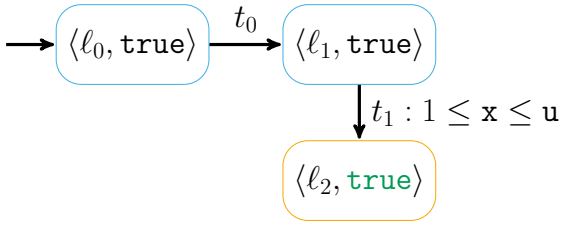
$$\overbrace{0 < \mathtt{y}}^{\text{guard}}, \overbrace{\mathtt{true}}^{\text{pre. label}} \xrightarrow{\text{update}} 0 < \mathtt{y} \qquad 0 < \mathtt{y} \not\models 0 \geq \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

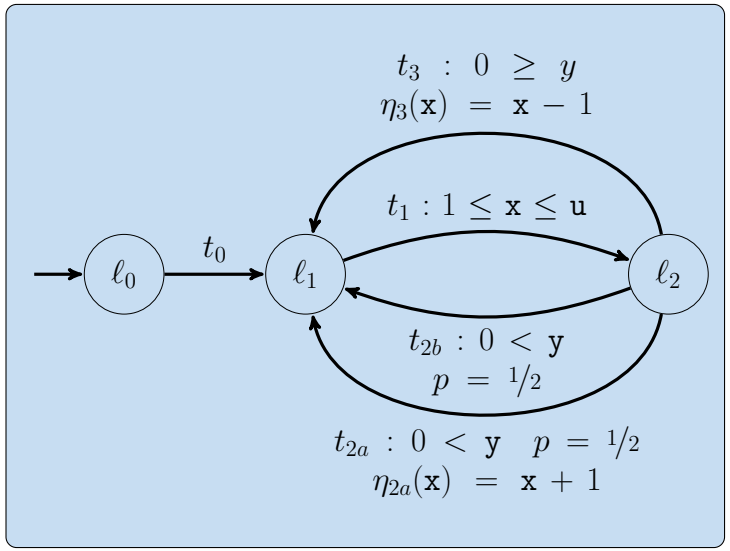
Apply t_{2a}:

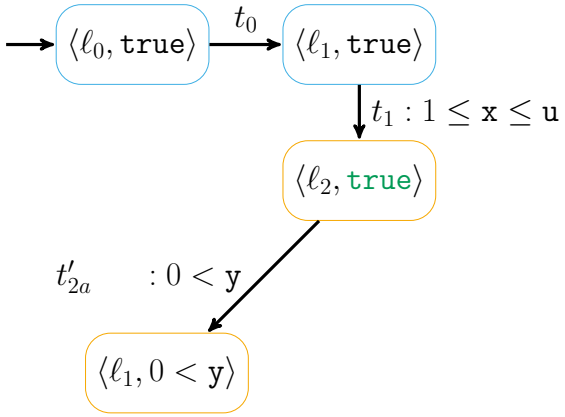
$$\overbrace{0 < \mathtt{y}}^{\text{guard}}, \overbrace{\mathtt{true}}^{\text{pre. label}} \xrightarrow{\text{update}} 0 < \mathtt{y} \qquad 0 < \mathtt{y} \models 0 < \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

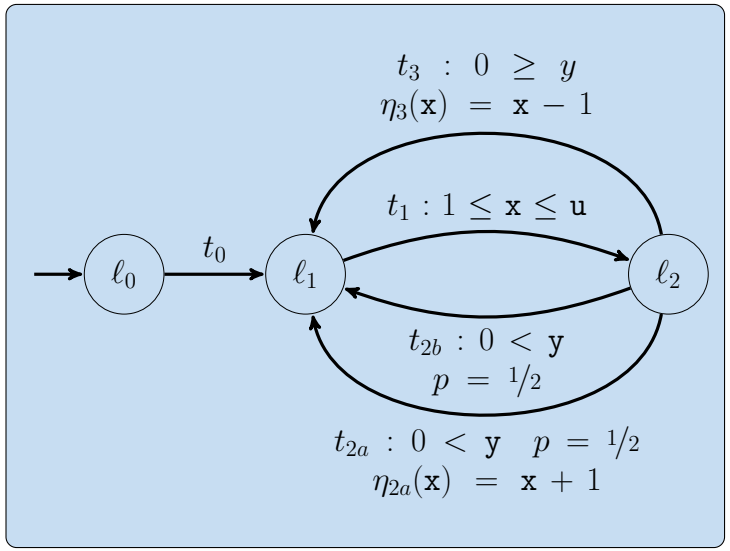
Apply t_{2a}:

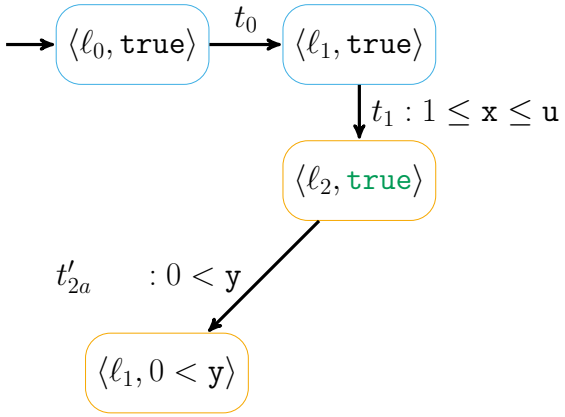
$$\overbrace{0 < \mathtt{y}}^{\text{guard}}, \overbrace{\mathtt{true}}^{\text{pre. label}} \xrightarrow{\text{update}} 0 < \mathtt{y} \qquad 0 < \mathtt{y} \models 0 < \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

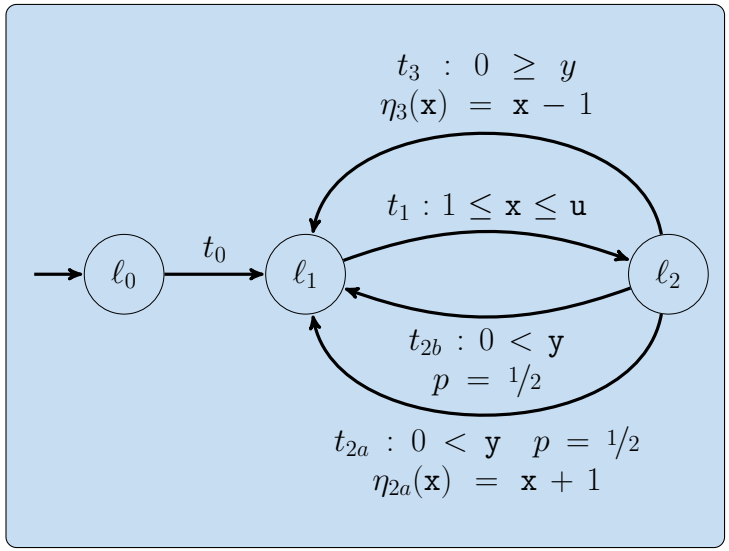
Apply t_{2b}:

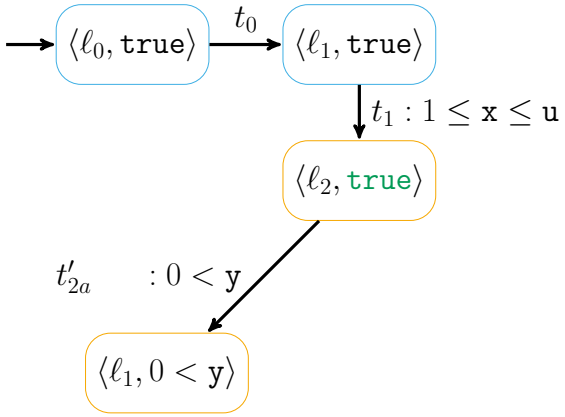
$$\overbrace{0 < \mathtt{y}}^{\text{guard}}, \overbrace{\mathtt{true}}^{\text{pre. label}} \xrightarrow{\text{update}} 0 < \mathtt{y} \qquad 0 < \mathtt{y} \not\models 0 \geq \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

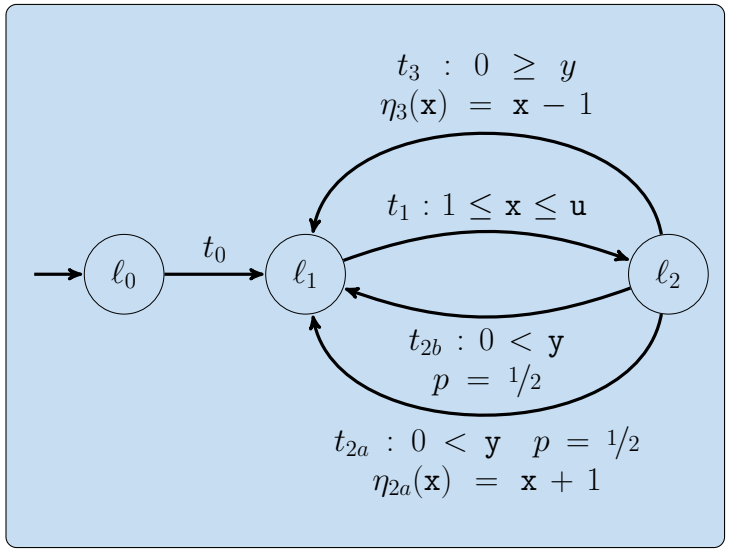
Apply t_{2b}:

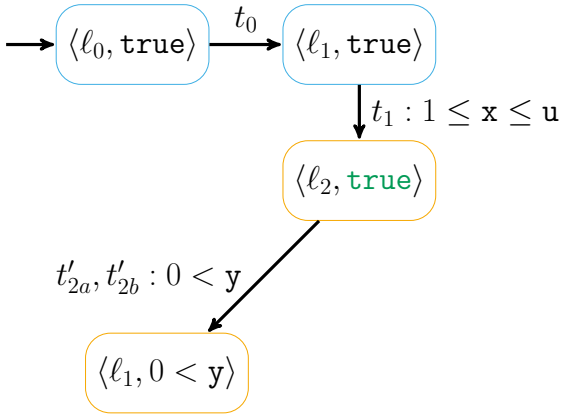
$$\overbrace{0 < \mathtt{y}}^{\text{guard}}, \overbrace{\mathtt{true}}^{\text{pre. label}} \xrightarrow{\text{update}} 0 < \mathtt{y} \qquad 0 < \mathtt{y} \models 0 < \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

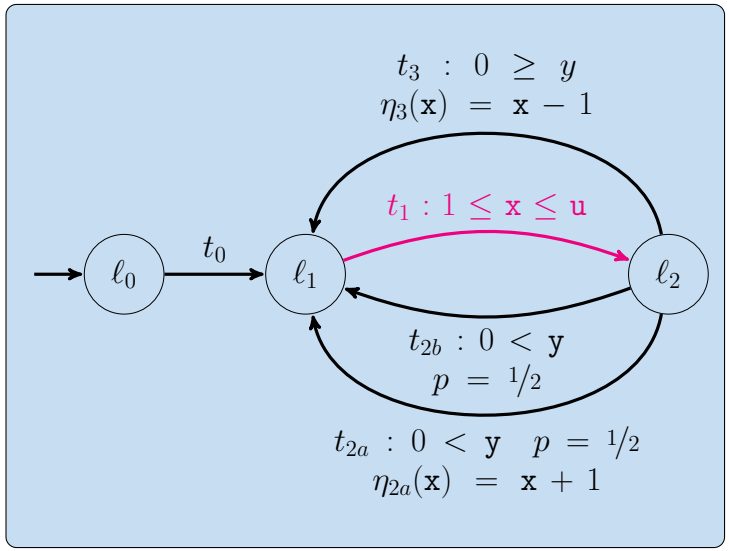
Apply t_{2b}:

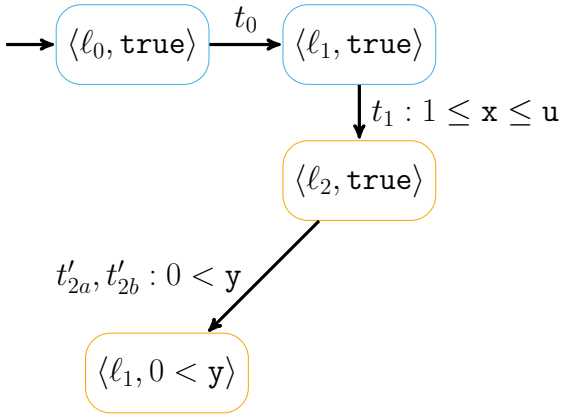
$$\underbrace{0 < \mathtt{y}}_{\text{guard}}, \underbrace{\mathtt{true}}_{\text{pre. label}} \xrightarrow{\text{update}} 0 < \mathtt{y} \qquad 0 < \mathtt{y} \models 0 < \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





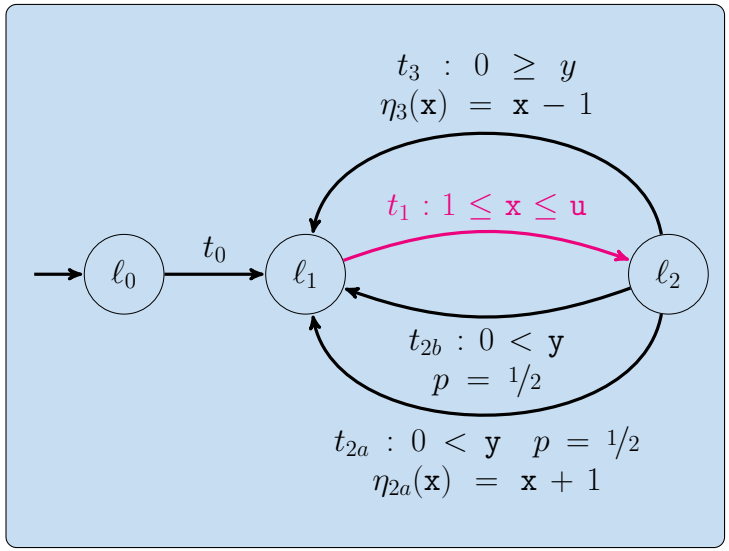
- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

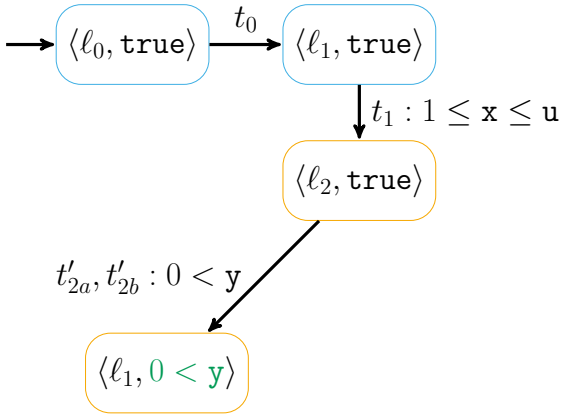
Apply t_1:

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

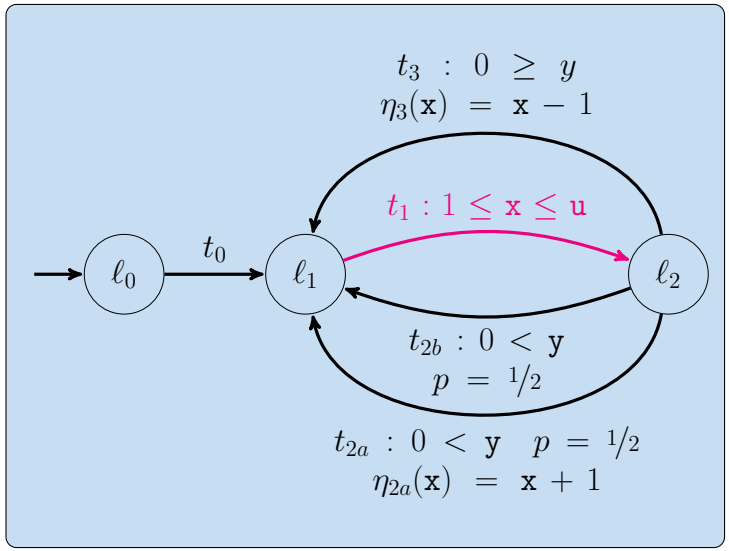
Apply t_1:

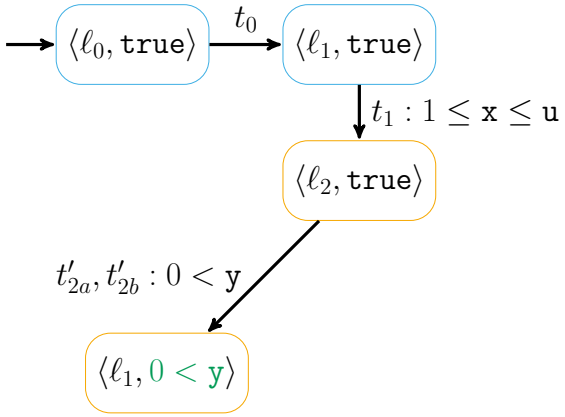
$$\overbrace{1 \leq \mathtt{x} \leq \mathtt{u}}^{\text{guard}}, \overbrace{0 < \mathtt{y}}^{\text{pre. label}} \xrightarrow{\text{update}} 1 \leq \mathtt{x} \leq \mathtt{u} \wedge 0 < \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

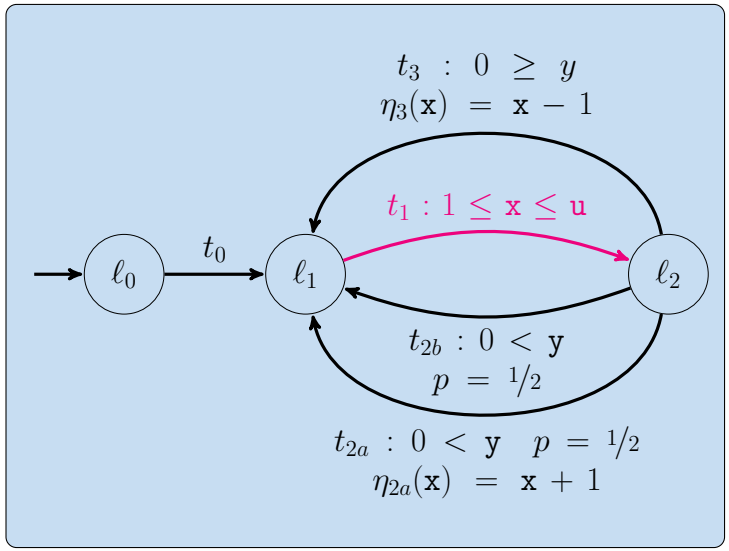
Apply t_1:

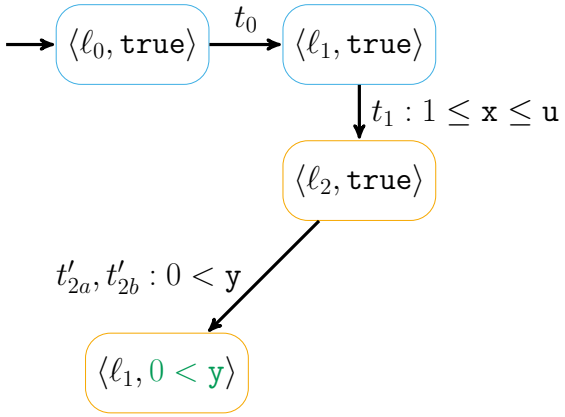
$$\overbrace{1 \leq \mathtt{x} \leq \mathtt{u}}^{\text{guard}}, \overbrace{0 < \mathtt{y}}^{\text{pre. label}} \xrightarrow{\text{update}} 1 \leq \mathtt{x} \leq \mathtt{u} \wedge 0 < \mathtt{y} \qquad 1 \leq \mathtt{x} \leq \mathtt{u} \wedge 0 < \mathtt{y} \not\models 0 \geq \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

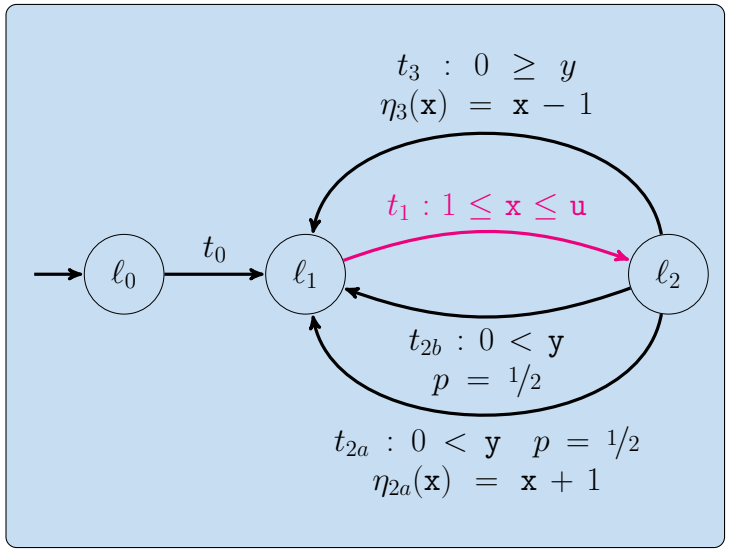
Apply t_1:

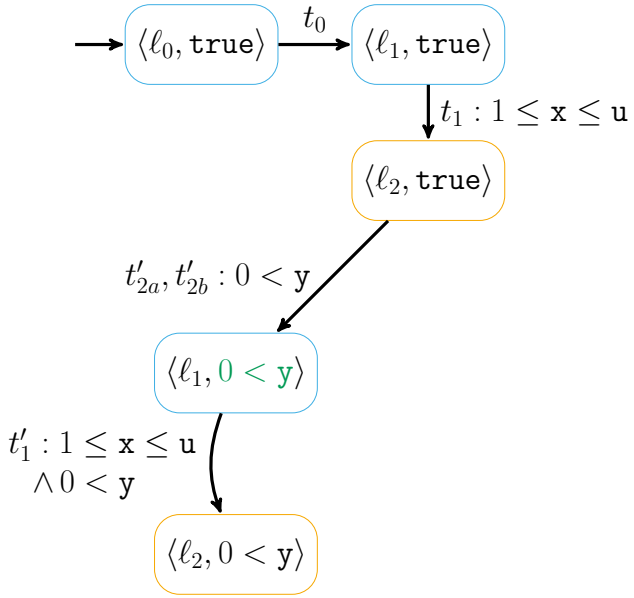
$$\overbrace{1 \leq \mathtt{x} \leq \mathtt{u}}^{\text{guard}}, \overbrace{0 < \mathtt{y}}^{\text{pre. label}} \xrightarrow{\text{update}} 1 \leq \mathtt{x} \leq \mathtt{u} \wedge 0 < \mathtt{y} \qquad 1 \leq \mathtt{x} \leq \mathtt{u} \wedge 0 < \mathtt{y} \models 0 < \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

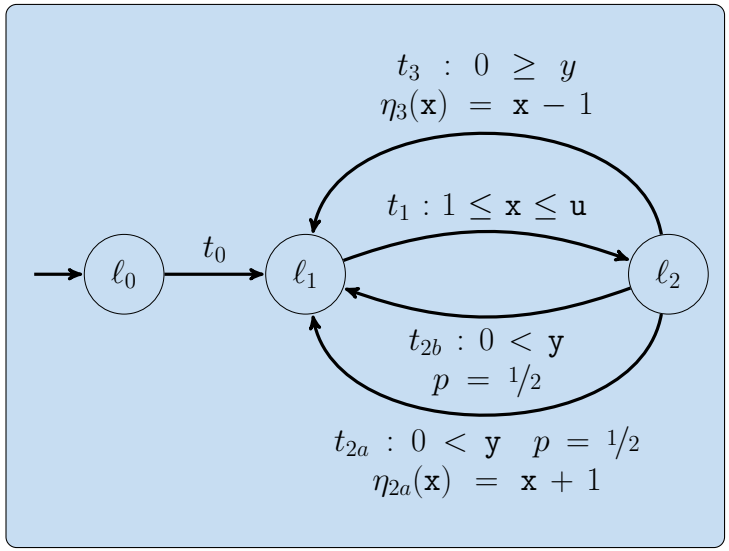
Apply t_1:

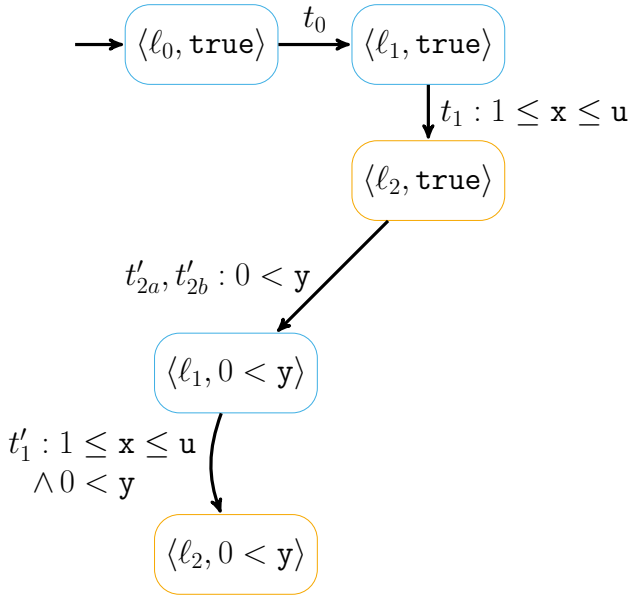
$$\overbrace{1 \leq \mathtt{x} \leq \mathtt{u}}^{\text{guard}}, \overbrace{0 < \mathtt{y}}^{\text{pre. label}} \xrightarrow{\text{update}} 1 \leq \mathtt{x} \leq \mathtt{u} \wedge 0 < \mathtt{y} \qquad 1 \leq \mathtt{x} \leq \mathtt{u} \wedge 0 < \mathtt{y} \models 0 < \mathtt{y}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations



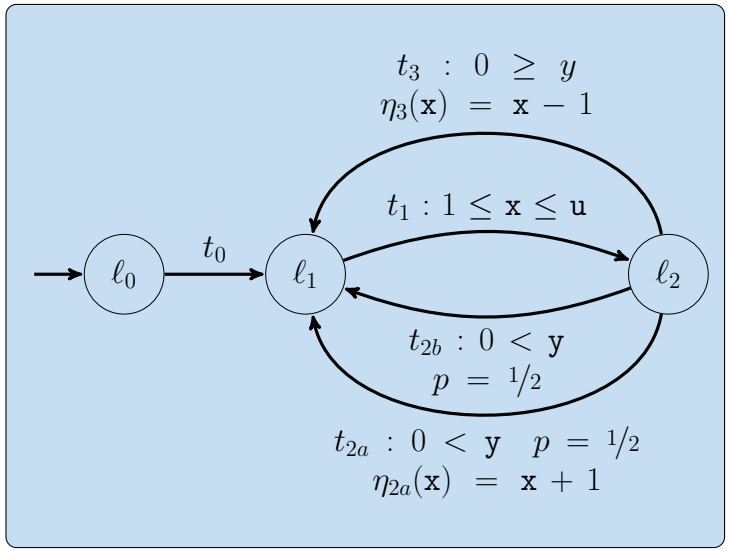


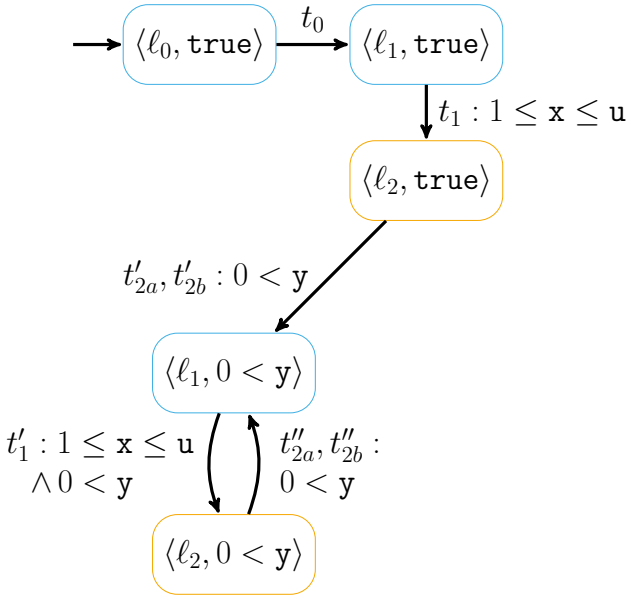
- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations



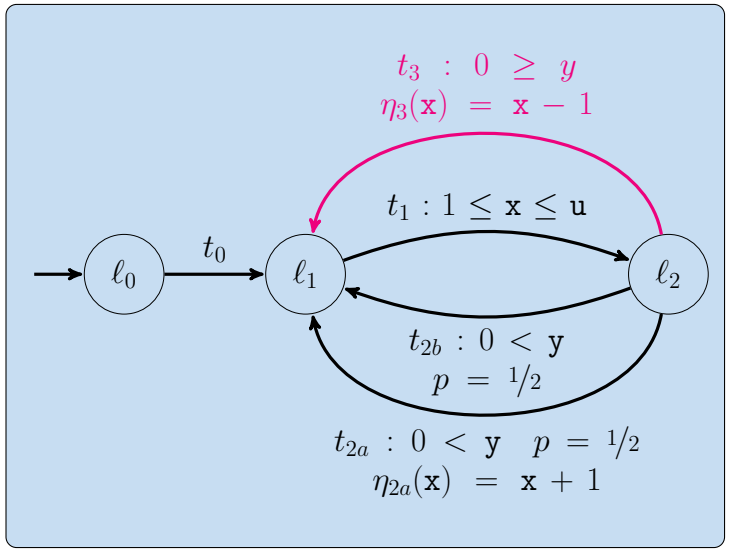


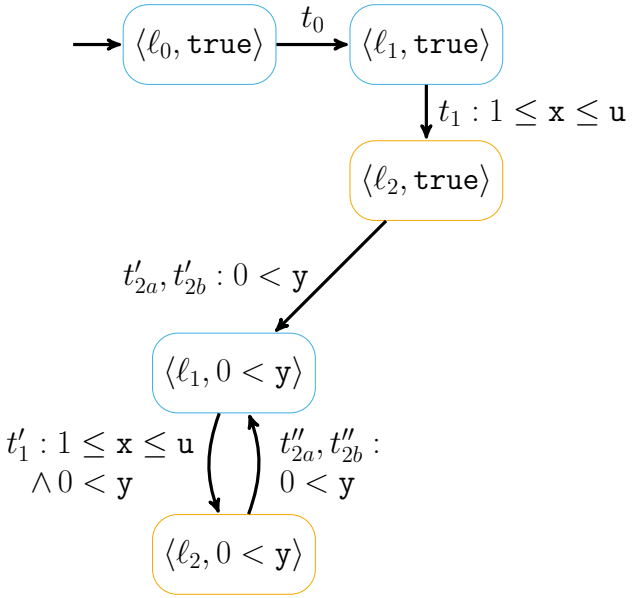
- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





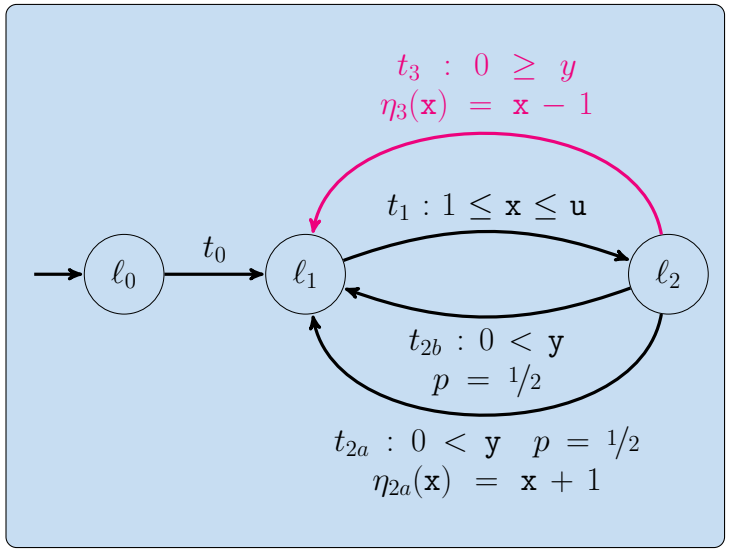
- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

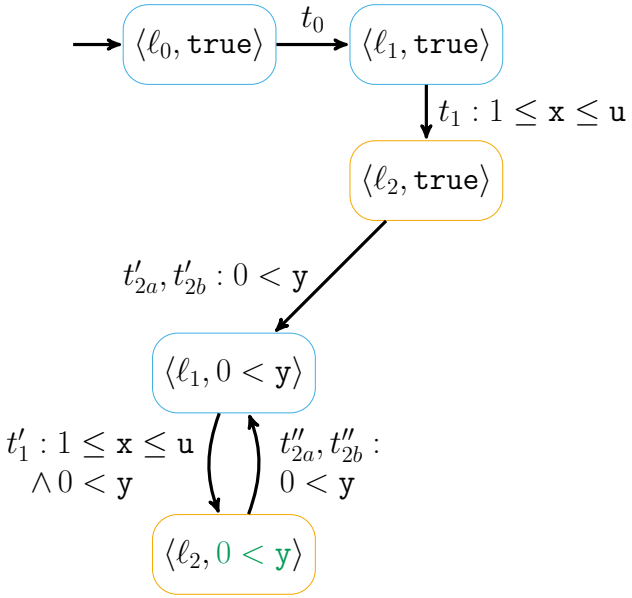
Apply t_3:

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

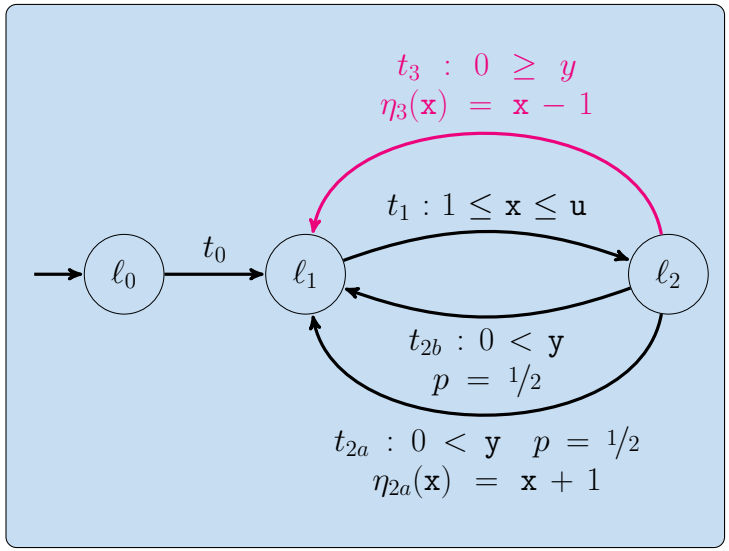
Apply t_3:

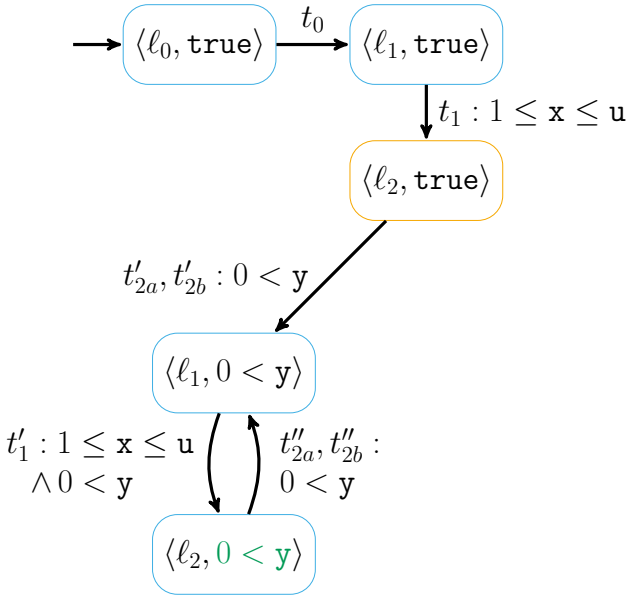
$$\overbrace{0 \geq \mathtt{y}}^{\text{guard}}, \overbrace{0 < \mathtt{y}}^{\text{pre. label}} \xrightarrow{\text{update}} \mathtt{false}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

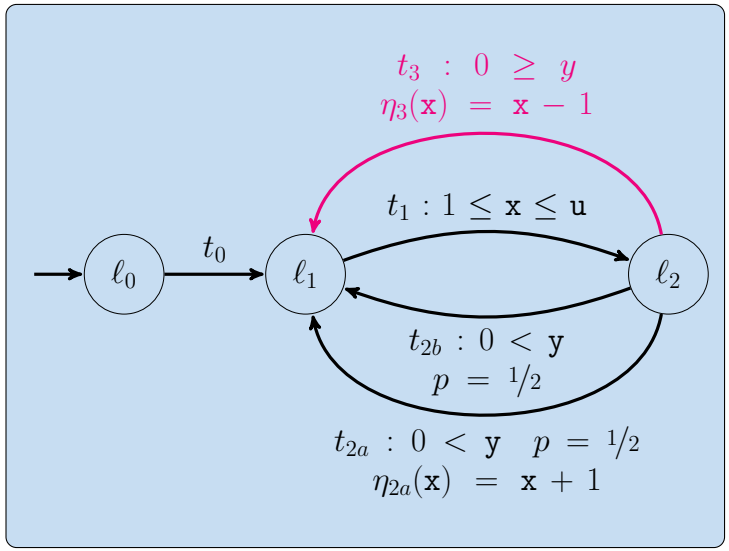
Apply t_3:

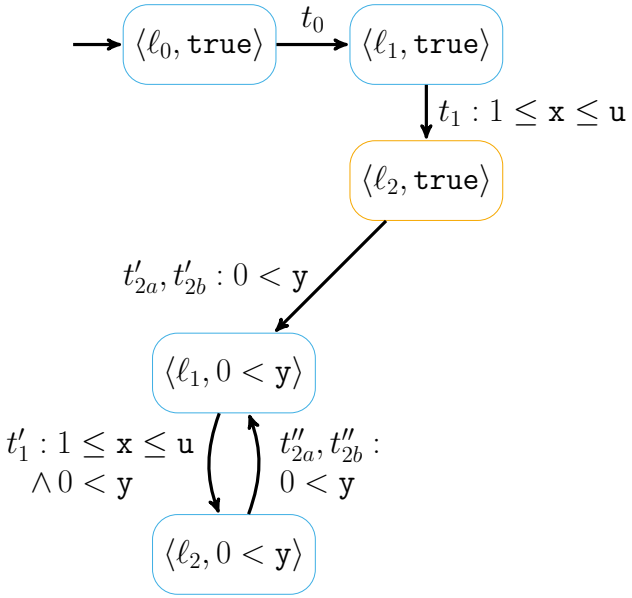
$$\overbrace{0 \geq \mathtt{y}}^{\text{guard}}, \overbrace{0 < \mathtt{y}}^{\text{pre. label}} \xrightarrow{\text{update}} \mathtt{false}$$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations



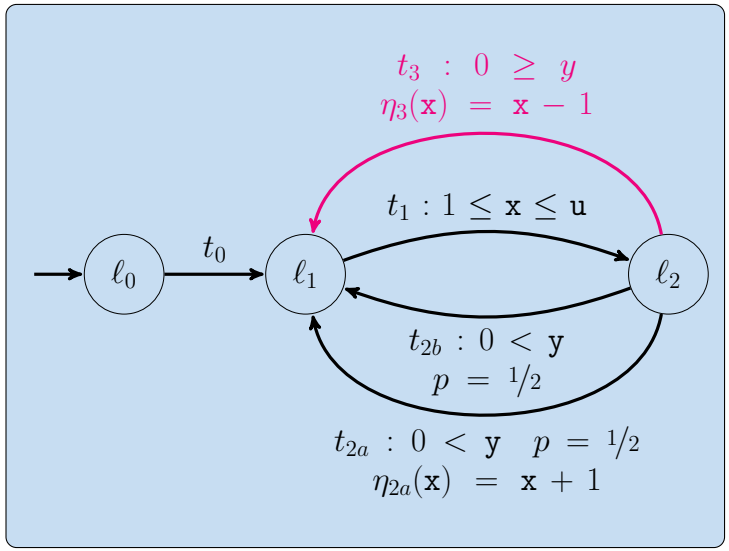


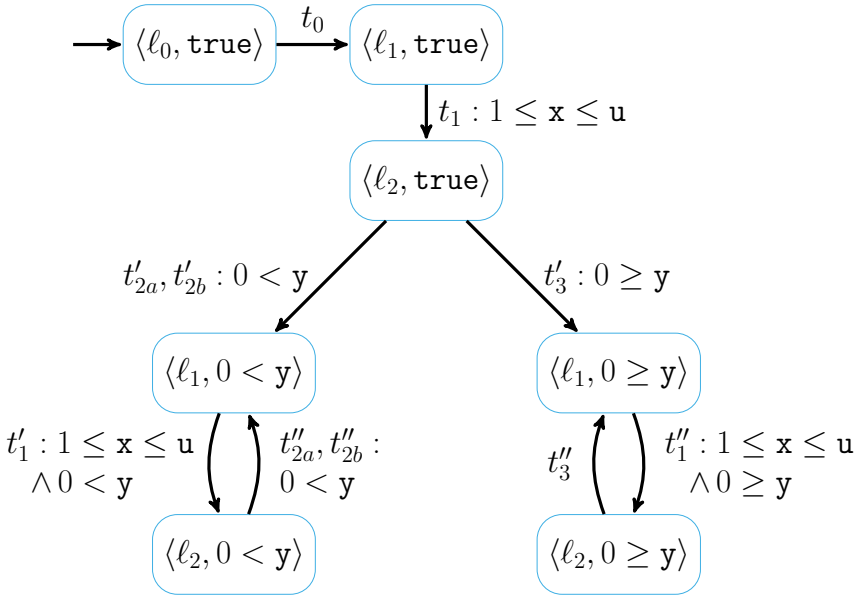
- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations



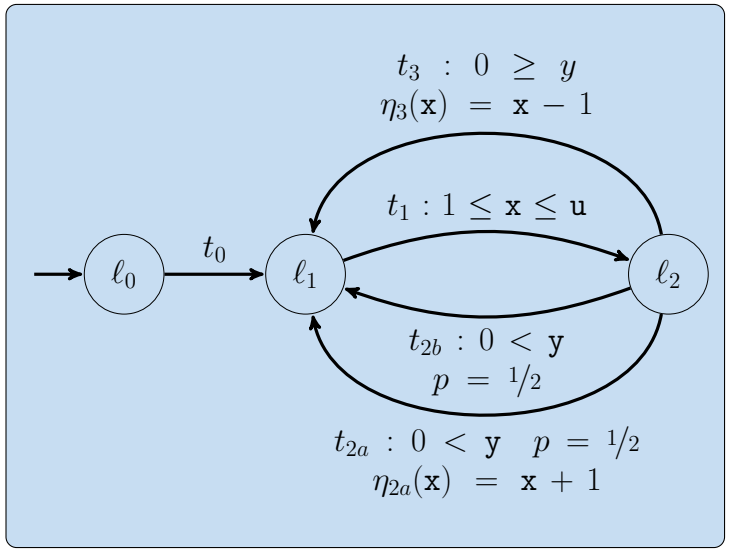


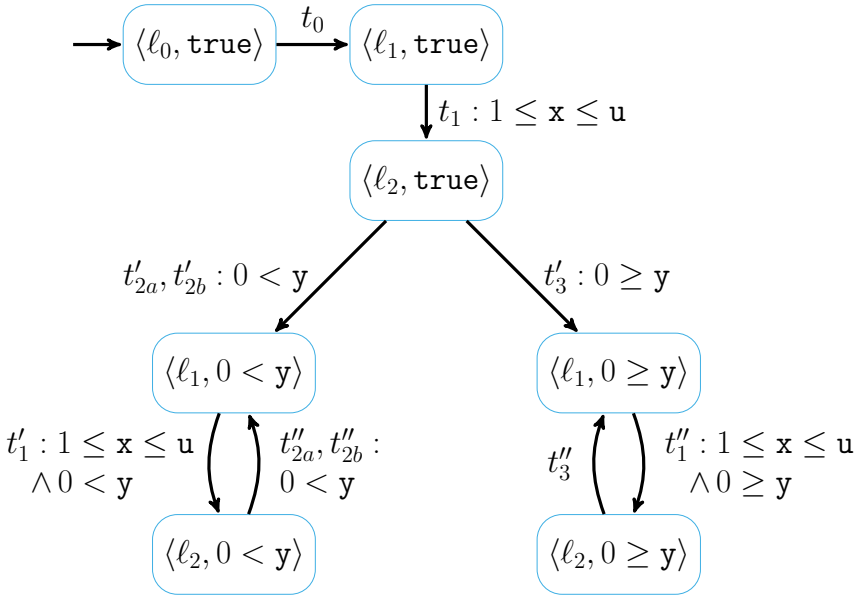
- Abstraction Layer: $\Psi = \{0 \geq \mathtt{y}, 0 < \mathtt{y}\}$ [Doménech et al.]

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations



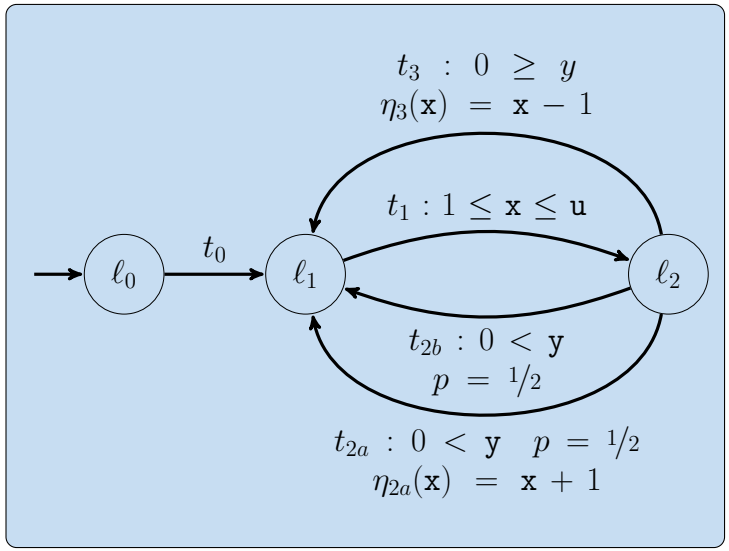


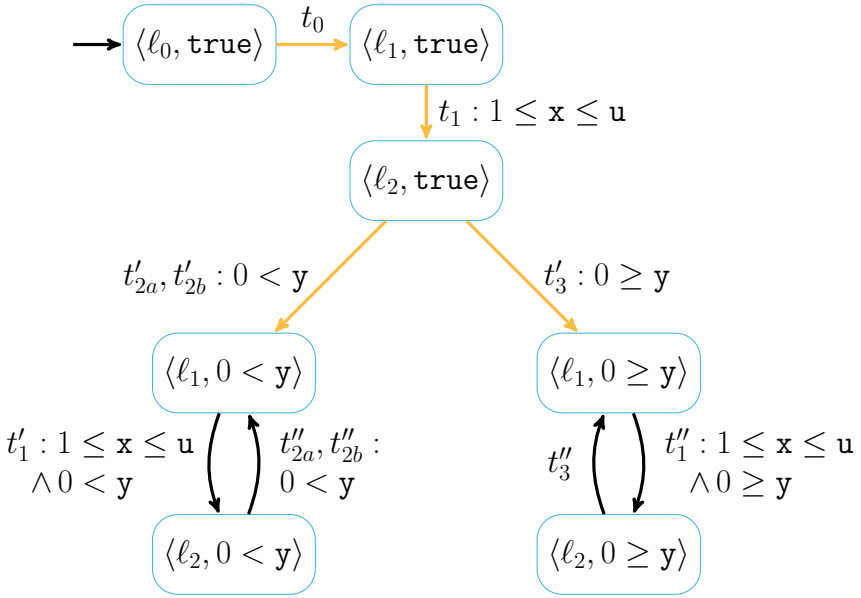
- Analyze runtime complexity:

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations



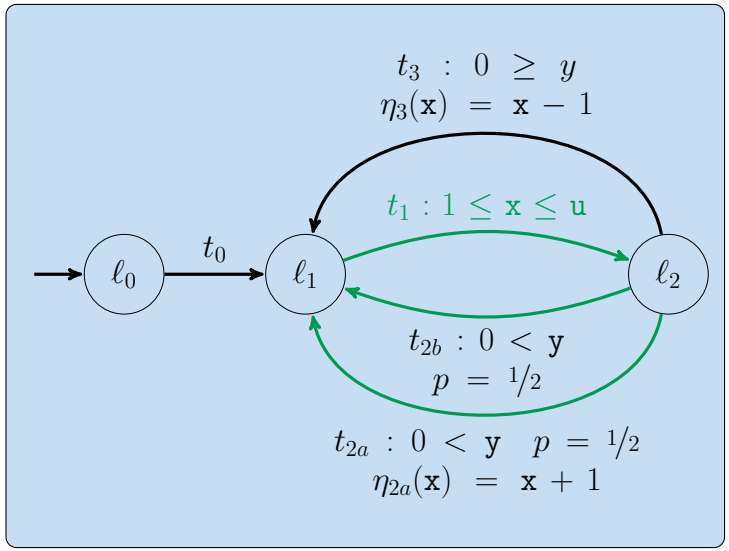


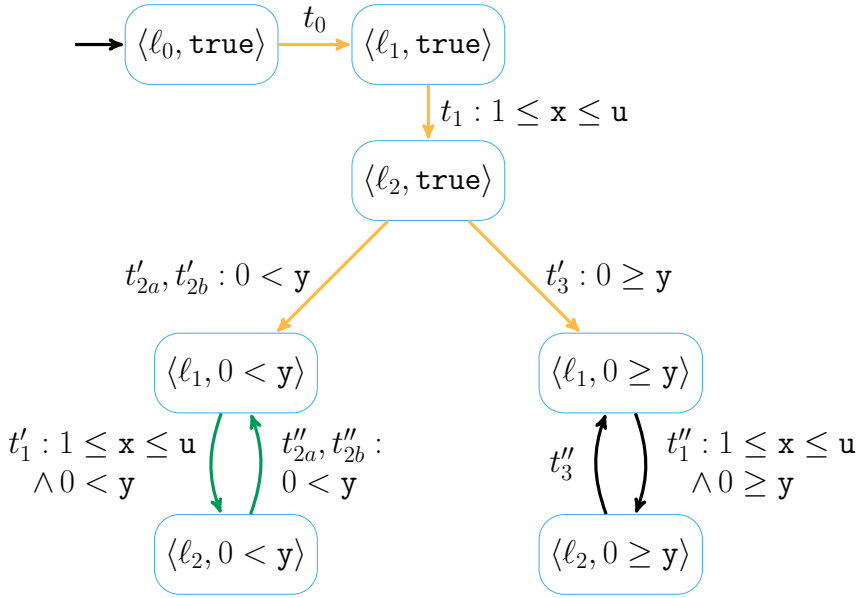
- Analyze runtime complexity:
  - non-cyclic transitions: runtime bound 1

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations



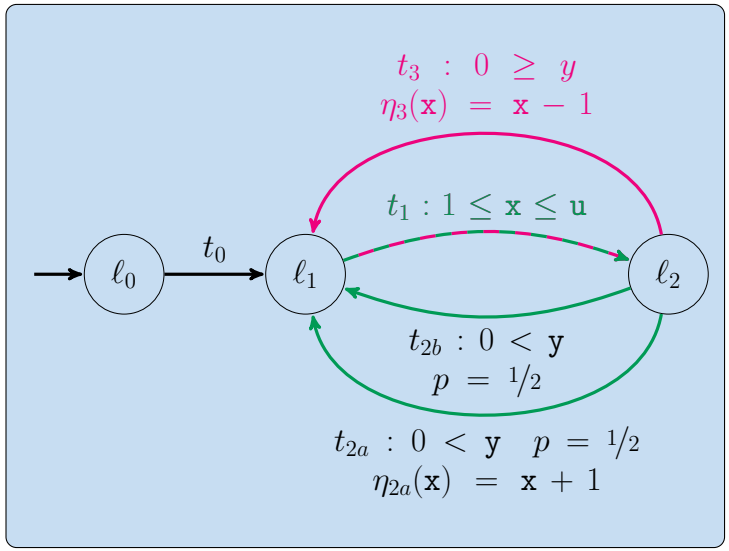


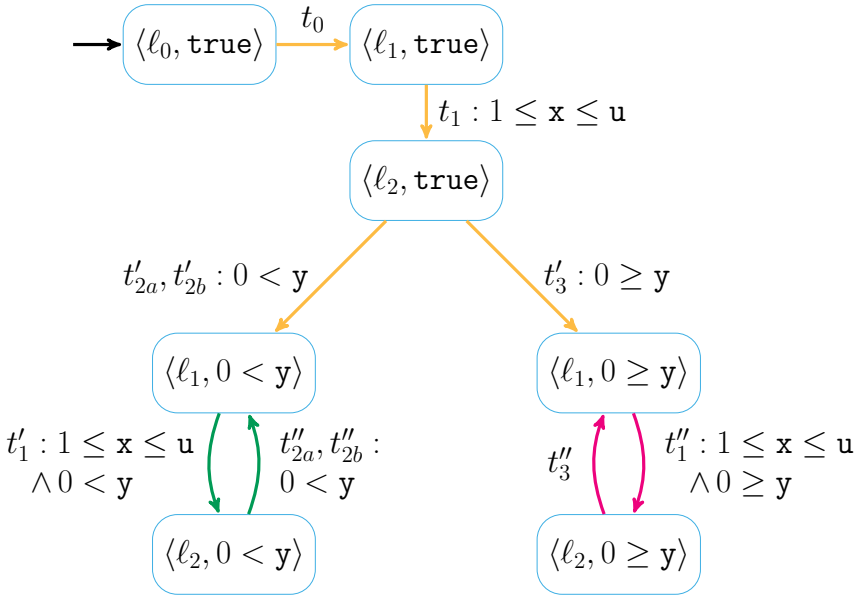
- Analyze runtime complexity:
  - non-cyclic transitions: runtime bound 1
  - transitions $t'_1, \{t''_{2a}, t''_{2b}\}$: expected runtime bound $2 \cdot \mathtt{u}$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

CFR via Partial Evaluation – Abstraction Layer

Idea: Use guards and updates to generate refined locations





- Analyze runtime complexity:
  - non-cyclic transitions: runtime bound 1
  - transitions $t'_1, \{t''_{2a}, t''_{2b}\}$: expected runtime bound $2 \cdot \mathtt{u}$
  - transitions t''_1, t''_3: runtime bound $\mathtt{u}$

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Complexity of CFR and Soundness

Complexity of CFR and Soundness

- Complexity:

Complexity of CFR and Soundness

- Complexity:
  - For abstraction layer Ψ maximal $2^{|\Psi|}$ locations

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Complexity of CFR and Soundness

- Complexity:
  - For abstraction layer Ψ maximal $2^{|\Psi|}$ locations
    $\Rightarrow$ modular CFR only on “not yet good enough” subprograms

Complexity of CFR and Soundness

- Complexity:
  - For abstraction layer Ψ maximal $2^{|\Psi|}$ locations
    $\Rightarrow$ modular CFR only on “not yet good enough” subprograms

- Soundness:

IJCAR ’24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Complexity of CFR and Soundness

- Complexity:
  - For abstraction layer Ψ maximal $2^{|\Psi|}$ locations
    $\Rightarrow$ modular CFR only on “not yet good enough” subprograms

- Soundness:
  - expected runtime of original program $\leq$ expected runtime of new program

IJCAR ’24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Complexity of CFR and Soundness

- Complexity:
  - For abstraction layer Ψ maximal $2^{|\Psi|}$ locations
    $\Rightarrow$ modular CFR only on “not yet good enough” subprograms

- Soundness:
  - expected runtime of original program $\leq$ expected runtime of new program
    Idea: Probability preserving bijections between paths

IJCAR ’24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Complexity of CFR and Soundness

- Complexity:
  - For abstraction layer Ψ maximal $2^{|\Psi|}$ locations
    $\Rightarrow$ modular CFR only on “not yet good enough” subprograms

- Soundness:
  - expected runtime of original program $\leq$ expected runtime of new program
    Idea: Probability preserving bijections between paths
  - expected runtime of original program $\geq$ expected runtime of new program

IJCAR ’24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Complexity of CFR and Soundness

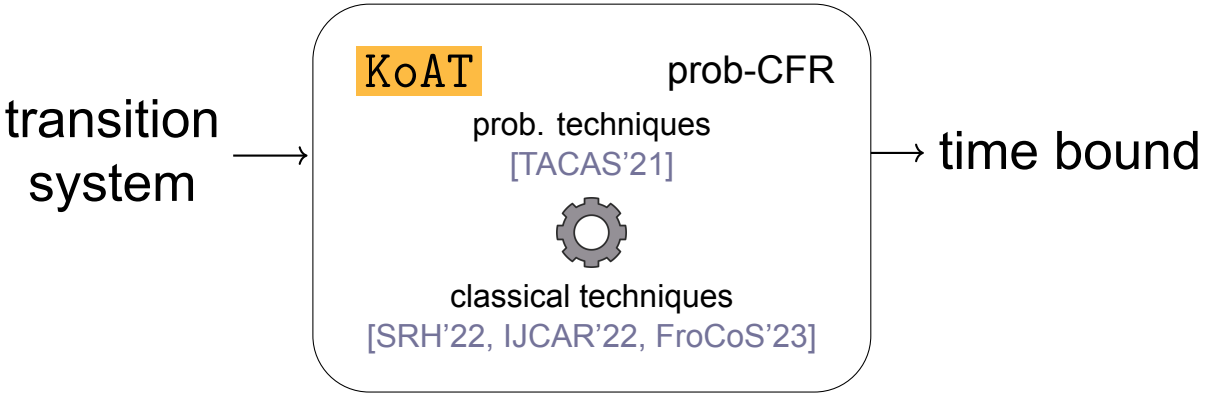
- Complexity:
  - For abstraction layer Ψ maximal $2^{|\Psi|}$ locations
    $\Rightarrow$ modular CFR only on “not yet good enough” subprograms

- Soundness:
  - expected runtime of original program $\leq$ expected runtime of new program
    Idea: Probability preserving bijections between paths
  - expected runtime of original program $\geq$ expected runtime of new program
    Idea: new non-determinism does not cause problems
    (Markovian Scheduler vs. History-Dependent Markovian Scheduler)

IJCAR ’24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Overview

Goal: Infer upper runtime bounds for (probabilistic) programs



Evaluation of our Implementation in KoAT

- Collected 90 benchmarks from the literature

	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{>2})$	$\mathcal{O}(EXP)$	$< \infty$	AVG(s)^+
`eco-imp`	8	35	6	0	0	49	0.34
`Absynth`	7	35	9	0	0	51	2.86

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Evaluation of our Implementation in KoAT

- Collected 90 benchmarks from the literature

	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{>2})$	$\mathcal{O}(EXP)$	$< \infty$	$\mathrm{AVG(s)}^+$
eco-imp	8	35	6	0	0	49	0.34
Absynth	7	35	9	0	0	51	2.86
KoAT	9	41	16	2	1	69	2.71

- KoAT: original probabilistic version of KoAT [TACAS'21]

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Evaluation of our Implementation in KoAT

- Collected 90 benchmarks from the literature

	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{>2})$	$\mathcal{O}(EXP)$	$< \infty$	AVG(s)^+
eco-imp	8	35	6	0	0	49	0.34
Absynth	7	35	9	0	0	51	2.86
KoAT	9	41	16	2	1	69	2.71
KoAT + CFR	11	56	14	2	1	84	9.97

- KoAT: original probabilistic version of KoAT [TACAS'21]
- KoAT + CFR: improved version with probabilistic CFR

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Evaluation of our Implementation in KoAT

- Collected 90 benchmarks from the literature

	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{>2})$	$\mathcal{O}(EXP)$	$< \infty$	$\mathrm{AVG(s)}^+$	succ. rate
eco-imp	8	35	6	0	0	49	0.34	54%
Absynth	7	35	9	0	0	51	2.86	56%
KoAT	9	41	16	2	1	69	2.71	76%
KoAT + CFR	11	56	14	2	1	84	9.97	93%

- KoAT: original probabilistic version of KoAT [TACAS'21]
- KoAT + CFR: improved version with probabilistic CFR
- KoAT + CFR solves 93% of benchmarks.

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Evaluation of our Implementation in KoAT

- Collected 90 benchmarks from the literature

	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{>2})$	$\mathcal{O}(EXP)$	$< \infty$	$\mathrm{AVG(s)}^+$	succ. rate
eco-imp	8	35	6	0	0	49	0.34	54%
Absynth	7	35	9	0	0	51	2.86	56%
KoAT	9	41	16	2	1	69	2.71	76%
KoAT + CFR	11	56	14	2	1	84	9.97	93%

- KoAT: original probabilistic version of KoAT [TACAS'21]
- KoAT + CFR: improved version with probabilistic CFR
- KoAT + CFR solves 93% of benchmarks.
- Implemented prob-CFR as an independent subsystem

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Evaluation of our Implementation in KoAT

- Collected 90 benchmarks from the literature

	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{>2})$	$\mathcal{O}(EXP)$	$< \infty$	$\mathrm{AVG(s)}^+$	succ. rate
eco-imp	8	35	6	0	0	49	0.34	54%
Absynth	7	35	9	0	0	51	2.86	56%
KoAT	9	41	16	2	1	69	2.71	76%
KoAT + CFR	11	56	14	2	1	84	9.97	93%

- KoAT: original probabilistic version of KoAT [TACAS'21]
- KoAT + CFR: improved version with probabilistic CFR
- KoAT + CFR solves 93% of benchmarks.
- Implemented prob-CFR as an independent subsystem

  $\Rightarrow$ Maybe other tools profit from prob-CFR

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Live-Demo

```
(GOAL EXPECTEDCOMPLEXITY)
(STARTTERM (FUNCTIONSYMBOLS l0))
(VAR X Y U)
(RULES
  l0(X, Y, U) -> l1(X, Y, U)
  l1(X, Y, U) -> l2(X, Y, U) :|: 1 <= X && X <= U
  l2(X, Y, U) -> 0.5:l1(X + 1, Y, U)
              :+: 0.5:l1(X    , Y, U) :|: 0 < Y
  l2(X, Y, U) -> l1(X - 1, Y, U) :|: 0 >= Y
)
```

https://koat.verify.rwth-aachen.de

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Conclusion

- Conclusion

Conclusion

- Conclusion
  - Introduced modular approach for *probabilistic* CFR

Conclusion

- Conclusion
  - Introduced modular approach for *probabilistic* CFR
  - CFR might split-up program into classical and probabilistic parts

Conclusion

- Conclusion
  - Introduced modular approach for *probabilistic* CFR
  - CFR might split-up program into classical and probabilistic parts
    - classical techniques

      [SRH'22, IJCAR'22, FroCoS'23]

Conclusion

- Conclusion
  - Introduced modular approach for *probabilistic* CFR
  - CFR might split-up program into classical and probabilistic parts
    - classical techniques
      [SRH'22, IJCAR'22, FroCoS'23]
    - probabilistic techniques
      [TACAS'21]

Conclusion

- Conclusion
  - Introduced modular approach for *probabilistic* CFR
  - CFR might split-up program into classical and probabilistic parts
    - classical techniques
      [SRH'22, IJCAR'22, FroCoS'23]
    - probabilistic techniques
      [TACAS'21]
  - CFR can be used as a black-box by other tools

Conclusion

- Conclusion
  - Introduced modular approach for *probabilistic* CFR
  - CFR might split-up program into classical and probabilistic parts
    - classical techniques
      [SRH'22, IJCAR'22, FroCoS'23]
    - probabilistic techniques
      [TACAS'21]
  - CFR can be used as a black-box by other tools
  - `KoAT` outperforms other state-of-the-art tools

Conclusion

- Conclusion
  - Introduced modular approach for *probabilistic* CFR
  - CFR might split-up program into classical and probabilistic parts
    - classical techniques
      [SRH'22, IJCAR'22, FroCoS'23]
    - probabilistic techniques
      [TACAS'21]
  - CFR can be used as a black-box by other tools
  - `KoAT` outperforms other state-of-the-art tools

`https://koat.verify.rwth-aachen.de/`

Conclusion

- Conclusion
  - Introduced modular approach for *probabilistic* CFR
  - CFR might split-up program into classical and probabilistic parts
    - classical techniques
      [SRH'22, IJCAR'22, FroCoS'23]
    - probabilistic techniques
      [TACAS'21]
  - CFR can be used as a black-box by other tools
  - `KoAT` outperforms other state-of-the-art tools

`https://koat.verify.rwth-aachen.de/`

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification

Conclusion

- Conclusion
  - Introduced modular approach for *probabilistic* CFR
  - CFR might split-up program into classical and probabilistic parts
    - classical techniques
      [SRH'22, IJCAR'22, FroCoS'23]
    - probabilistic techniques
      [TACAS'21]
  - CFR can be used as a black-box by other tools
  - `KoAT` outperforms other state-of-the-art tools

`https://koat.verify.rwth-aachen.de/`

Thank You!

IJCAR '24
Nils Lommen, Éléanore Meyer, and Jürgen Giesl
RWTH Aachen University – Programming Languages and Verification